08

Mãos à obra!

Vamos criar o arquivo chamado `contatos_utils.py`. Neste arquivo, vamos importar os módulos `csv`, `pickle` e `json`, além de importar a classe `Contato` do módulo `contato`.

```
import csv, pickle, json
from contato import Contato
```

Vamos começar pela função que traz os dados do `csv` para o objetos Python. Nessa função abriremos o arquivo utilizando o `with` e invocaremos a função `reader()` do módulo `csv` passando o arquivo aberto como parâmetro.

A partir disso, criamos um novo contato e adicionamos a uma lista que será retornada no final da função:

```
def csv_para_contatos(caminho, encoding='latin_1'):
    contatos = []

    with open(caminho, encoding=encoding) as arquivo:
        leitor = csv.reader(arquivo)

        for linha in leitor:
            id, nome, email = linha

            contato = Contato(id, nome, email)
            contatos.append(contato)

return contatos
```

Bacana! Podemos testar essa função no arquivo `principal.py`, lembrando que precisamos importar o módulo `contato_utils` no começo do arquivo:

```
import contatos_utils

try:
    contatos = contatos_utils.csv_para_contatos('dados/contatos.csv')

    for contato in contatos:
        print(f'{contato.id} - {contato.nome} - {contato.email}')
except FileNotFoundError:
    print('Arquivo não encontrado')
except PermissionError:
print('Sem permissão de escrita')
```

Quando executamos o arquivo principal, vemos que os contatos são imprimidos na tela.

De volta ao arquivo `contatos_utils.py`, vamos escrever as funções que convertem a lista de contatos para serializáveis Python, com o `pickle`. Portanto:

```
def contatos_para_pickle(contatos, caminho):
    with open(caminho, mode='wb') as arquivo:
        pickle.dump(contatos, arquivo)

def pickle_para_contatos(caminho):
    with open(caminho, mode='rb') as arquivo:
        contatos = pickle.load(arquivo)

    return contatos
```

Abrimos a função como o modo binário (`b`), pois os arquivos `pickle` são binários. Na primeira função salvamos a lista no banco e na segunda efetuamos a leitura desses arquivos. Podemos, então, testar essas funções no arquivo `principal.py` :

```
import contatos_utils

try:
    contatos = contatos_utils.csv_para_contatos('dados/contatos.csv')

    contatos_utils.contatos_para_pickle(contatos, 'dados/contatos.pickle')
    contatos = contatos_utils.pickle_para_contatos('dados/contatos.pickle')

    for contato in contatos:
        print(f'{contato.id} - {contato.nome} - {contato.email}')
except FileNotFoundError:
    print('Arquivo não encontrado')
except PermissionError:
    print('Sem permissão de escrita')
```

Podemos ver que os contatos são listados corretamente, ou seja, as funções estão serializando e desserializando corretamente a lista de contatos.

Por fim, vamos criar as funções que serializam os objetos em JSON, para isso, vamos utilizar a função `dump()` do módulo `json` e passar como parâmetro `default` a função que realiza a conversão dos objetos do tipo contato:

```
def contatos_para_json(contatos, caminho):
    with open(caminho, mode='w') as arquivo:
        json.dump(contatos, arquivo, default=_contato_para_json)

def _contato_para_json(contato):
    return contato.__dict__
```

Na função que desserializa os objetos de JSON para contatos Python, criamos uma lista e vamos adicionando cada linha deste contato na lista. Como a função `load()` devolve um dicionário em que cada chave é um atributo do objeto, podemos desempacotar esse dicionário no construtor da classe `Contato` :

```
def json_para_contatos(caminho):
    contatos = []

    with open(caminho) as arquivo:
```

```
        contatos_json = json.load(arquivo)

        for contato in contatos_json:
            c = Contato(**contato)
            contatos.append(c)

    return contatos
```

No arquivo principal, vamos realizar a chamada das duas funções:

```
import contatos_utils

try:
    contatos = contatos_utils.csv_para_contatos('dados/contatos.csv')

    # contatos_utils.contatos_para_pickle(contatos, 'dados/contatos.pickle')
    # contatos = contatos_utils.pickle_para_contatos('dados/contatos.pickle')

    contatos_utils.contatos_para_json(contatos, 'dados/contatos.json')
    contatos = contatos_utils.json_para_contatos('dados/contatos.json')

    for contato in contatos:
        print(f'{contato.id} - {contato.nome} - {contato.email}')
except FileNotFoundError:
    print('Arquivo não encontrado')
except PermissionError:
    print('Sem permissão de escrita')
```

Ao final, o arquivo `contato_utils.py` fica parecido com este:

```
import csv, pickle, json
from contato import Contato


def csv_para_contatos(caminho, encoding='latin_1'):
    contatos = []

    with open(caminho, encoding=encoding) as arquivo:
        leitor = csv.reader(arquivo)

        for linha in leitor:
            id, nome, email = linha

            contato = Contato(id, nome, email)
            contatos.append(contato)

    return contatos


def contatos_para_pickle(contatos, caminho):
    with open(caminho, mode='wb') as arquivo:
        pickle.dump(contatos, arquivo)

def pickle_para_contatos(caminho):
    with open(caminho, mode='rb') as arquivo:
```

```
            contatos = pickle.load(arquivo)

        return contatos


    def contatos_para_json(contatos, caminho):
        with open(caminho, mode='w') as arquivo:
            json.dump(contatos, arquivo, default=_contato_para_json)

    def _contato_para_json(contato):
        return contato.__dict__

    def json_para_contatos(caminho):
        contatos = []

        with open(caminho) as arquivo:
            contatos_json = json.load(arquivo)

            for contato in contatos_json:
                c = Contato(**contato)
                contatos.append(c)

        return contatos
```