Introdução ao Python

Manipulação de arquivos

Henrique Y. Shishido

Departamento de Computação
Universidade Tecnológica Federal do Paraná

Introdução

- A manipulação de arquivos é uma atividade frequente em aplicações de inteligência artificial para a leitura de dados.
- Para este contexto, o Python oferece funções para criação, leitura, atualização e exclusão de arquivos.

Manipulação de arquivos

- A principal função para trabalhar com arquivos é a `open()`.
- A função `open()` necessita dois parâmetros: o nome do arquivo e o modo de acesso. Há quatro modos de acesso:
  - `"r"` - leitura (valor padrão) - abre um arquivo para leitura. Em caso de inexistência do arquivo, retorna um erro.
  - "a" - adição - abre um arquivo para a adição de mais conteúdo. Cria um arquivo caso ele não exista.
  - `"w"` - escrita - abre um arquivo para a escrita. Cria um arquivo caso ele não exista.
  - `"x"` - criação - cria um arquivo específico e retorna um erro caso o arquivo já exista.
- Além disso, é preciso especificar se o arquivo será tratado como binário ou texto.
  - `"t"` - texto - valor padrão. Modo Texto.
  - `"b"` - binário - Modo binário (imagens, vídeo, áudio, etc.)

Sintaxe

- Para abrir um arquivo para leitura, somente especificar o seu nome é suficiente.

```
f = open("meuArquivo.txt")
```

- O comando acima é equivalente a:

```
f = open("meuArquivo.txt", "rt")
```

Leitura de conteúdo

Listing 1: meuArquivo.txt

```
O mundo tem vários países.
Cada país possui estados.
Cada estado possui cidades.
```

- A função `open()` retorna um objeto de arquivo que pode ser lido por meio do método `read()`:

```
>>> f = open("meuArquivo.txt")
>>> print(f.read())
O mundo tem vários países.
Cada país possui estados.
Cada estado possui cidades.
```

Leitura de arquivo em diferentes localizações

- Se um arquivo não estiver no mesmo diretório onde o programa Python está sendo executado, é possível especificar o caminho do arquivo, conforme segue:

```
>>> f = open("/home/henriqueshishido/Documents/meuArquivo.txt")
```

Leitura de partes do arquivo

- Por padrão, a função `read()` retorna o texto na íntegra, mas pode-se especificar **quantos caracteres serão lidos**:

```
>>> f = open("meuArquivo.txt", "r")
>>> print(f.read(7))
O mundo
```

- Pode-se ler uma linha usando o método `readline()`:

```
>>> f = open("meuArquivo.txt", "r")
>>> print(f.readline())
O mundo tem vários países.
```

```
>>> f = open("meuArquivo.txt", "r")
>>> print(f.readline())
>>> print(f.readline())
O mundo tem vários países.
Cada país possui estados.
```

Leitura de todas as linhas de um arquivo

- Para realizar a leitura de todas as linhas de um arquivo, pode-se utilizar o comando `for` para iterar em cada linha do arquivo.

```
>>> f = open("meuArquivo.txt", "r")
>>> for linha in f:
>>>     print(linha)
O mundo tem vários países.
Cada país possui estados.
Cada estado possui cidades.
```

Fechando arquivos

- É uma boa prática sempre fechar arquivos após o seu uso, pois em alguns casos, as alterações realizadas em um arquivo só entrarão em vigor após o seu fechamento.

```
>>> f = open("meuArquivo.txt", "r")
>>> print(f.readline())
>>> f.close()
```

Escrita em arquivo

- Para escrever em um arquivo existente, deve-se adicionar um parâmetro para a função `open()`:
  - `"a"` - append - irá adicionar novo conteúdo ao final do arquivo
  - `"w"` - escrtita - irá sobrescrever o conteúdo existente

```
>>> f = open("meuArquivo.txt", "a")
>>> f.write("Cada cidade possui várias ruas.")
>>> f.close()

>>> f = open("meuArquivo.txt", "r")
>>> print(f.read())
O mundo tem vários países.
Cada país possui estados.
Cada estado possui cidades.
Cada cidade possui várias ruas.
```

Escrita em arquivo

- Para escrever em um arquivo existente, deve-se adicionar um parâmetro para a função `open()`:
  - `"a"` - append - irá adicionar novo conteúdo ao final do arquivo
  - `"w"` - escrtita - irá sobrescrever o conteúdo existente

```
>>> f = open("meuArquivo.txt", "w")
>>> f.write(Cada cidade possui várias ruas.")
>>> f.close()

>>> f.open("meuArquivo.txt", "r")
>>> print(f.read())
Cada cidade possui várias ruas.
```

Criando um novo arquivo

- Para criar um novo arquivo em Python, pode-se usar a função `open()` com um dos parâmetros a seguir:
  - `"x"` - create - cria um arquivo, retorna um erro se o arquivo já existir;
  - `"a"` - append - cria um arquivo se o arquivo não existir
  - `"w"` - cria um arquivo se o arquivo não existir

```
#Considerando que o arquivo existente meuArquivo.txt
>>> f = open("meuArquivo.txt", "x")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
FileExistsError: [Errno 17] File exists: 'meuArquivo.txt'
```

Excluindo um arquivo

- Para excluir um arquivo, deve-se importar o módulo os e invocar a função `os.remove()`:

```
>>> import os
>>> os.remove("meuArquivo.txt")
```

- Entretanto, caso o arquivo a ser removido não exista, o Python irá produzir um erro. Por isso, é recomendado a verificação da existência do arquivo antes de removê-lo:

```
>>> import os
>>> if os.path.exists("meuArquivo.txt"):
>>>     os.remove("meuArquivo.txt")
>>> else:
>>>     print("O arquivo não existe")
```

Excluindo pastas

- Para excluir uma pasta, pode-se utilizar o método `os.rmdir()`. Só é possível excluir uma pasta se ela estiver vazia.

```
>>> import os
>>> if os.path.exists("minhaPasta"):
>>>     os.rmdir("minhaPasta")
>>> else:
>>>     print("Pasta não existe!")
```

- Em casos em que a pasta a ser excluída não está vazia, pode-se utilizar o método `rmtree()` do módulo `shutil`:

```
>>> import shutil
>>> shutil.rmtree("minhaPasta")
```