Estácio

EDUCAR PARA TRANSFORMAR

Desenv. Rápido de Aplicações em Python

Prof. Msc. Henrique Mota
profhenriquemota@gmail.com

Março de 2023

EDUCAR PARA TRANSFORMAR

Manipulando arquivos

Aula 03

Introdução

EDUCAR PARA TRANSFORMAR

- Saber manipular arquivos é de grande utilidade.
- Seja para armazenar resultados de um determinado script, para criar backups, consumir uma lista de um planilha, etc.
- A maioria das linguagens possuem meios para manipulação.
- Abordaremos formas de ler, criar e gravar arquivos em unidades de armazenamento.

Introdução

EDUCAR PARA TRANSFORMAR

- **__file__ (“dunder file”)** – constante que representa o caminho absoluto para o arquivo corrente.

- O módulo “os” fornece uma maneira simples de usar funcionalidades que são dependentes de sistema operacional. ex.: ler ou escrever um arquivo.

  - open(): utilizado para abrir arquivo em modos diferentes.
  - path(): utilizado para manipular estruturas de diretórios.

Módulo “os.path”

EDUCAR PARA TRANSFORMAR

- Implementa funções úteis em nomes de caminho de arquivos.
- Os parâmetros de path podem ser passados como strings ou bytes.
- Comumente nomes de arquivos representados como strings de caracteres (Unicode).

Módulo “os.path”

EDUCAR PARA TRANSFORMAR

- os.path.abspath(path): Retorna uma versão normalizada e absoluta do nome de caminho path.

- os.path.basename(path): Retorna o nome do arquivo base do caminho path. Quando executado com a path apenas até o diretório, “/foo/bar/” retorna uma string vazia (”).

- os.path.dirname(path): Retorna o nome do diretório do caminho path.

- os.path.exists(path): Retorna True se path se referir a um caminho existente ou um descritor de arquivo aberto. Retorna False para links simbólicos quebrados.

Módulo “os.path”

EDUCAR PARA TRANSFORMAR

- os.path.isabs(path): Retorna True se path for um nome de caminho absoluto.
- os.path.isfile(path): Retorna True se path for um arquivo regular existente.
- os.path.isdir(path): Retorna True se path for um diretório existente.
- os.path.islink(path): Retorna True se path se referir a uma entrada de diretório existente que é um link simbólico.
- os.path.join(path, *paths): Junta um ou mais componentes do caminho de forma inteligente.


hands
on

```python
import os
from os import path

print("arquivo corrente: ", __file__)
print("caminho absoluto: ", path.abspath(__file__))
print("nome do arquivo: ", path.basename(__file__))
print("nome do arquivo (em branco): ", path.basename("/Users/henriquemota/"))
print("string em branco: ", path.basename("/Users/henriquemota/"))
print("nome do diretório: ", path.dirname(__file__))
print("diretório existe: ", path.exists("/Users/henriquemota/"))
print("caminho absoluto: ", path.isabs(__file__))
print("é arquivo?'': ", path.isfile(__file__))
print("é diretório?: ", path.isdir(__file__))
print("é link simbólico?: ", path.islink(__file__))
BASE_DIR = path.join(path.dirname(__file__), "..", "aula_02")
print("diretório: ", BASE_DIR)

```

Abrindo arquivos

EDUCAR PARA TRANSFORMAR

- Para abrir um arquivo de texto utilizar a função open.

- A sintaxe padrão da função é open(nome, modo, buffering),
  - "nome" - o arquivo que será aberto
  - "modo" - a forma de abertura do arquivo
  - "buffering" - é quantidade de bytes reservados na memória para a abertura do arquivo (opcional).

- Também pode ser utilizada para criar novos arquivos. O que diferencia é o valor inserido no campo "modo".

- Os valores de modo e buffering são opcionais, quando não informados são adotados os valores padrões (modo – r; buffering valor definido pelo SO).

Abrindo arquivos

EDUCAR PARA TRANSFORMAR

Modo	Descrição
r	somente leitura
w	escrita (caso o arquivo já exista, ele será apagado e um novo arquivo vazio será criado)
a	leitura e escrita (adiciona o novo conteúdo ao fim do arquivo)
r+	leitura e escrita
w+	escrita (o modo w+, assim como o w, também apaga o conteúdo anterior do arquivo)
a+	leitura e escrita (abre o arquivo para atualização)

```
1  import os
2  from os import path
3
4  BASE_DIR = path.join(path.dirname(__file__), "files")
5  file = open(path.join(BASE_DIR, "file.txt"), "a")
6
```

Lendo arquivos

EDUCAR PARA TRANSFORMAR

- No exemplo anterior, o script tentará abrir (ou criar) um arquivo chamado “file.txt”.

- Uma vez aberto, podemos realizar a leitura do arquivo usando as funções:

  - read(n) - lê até “n” bytes, caso o valor não seja informado, realiza a leitura do arquivo inteiro.
  - readline( ) - retorna uma string contendo a linha corrente do arquivo.
  - readlines( ) - retorna uma lista de strings, sendo cada elemento uma linha do arquivo.

```
1   import os
2   from os import path
3
4   BASE_DIR = path.join(path.dirname(__file__), "files")
5   file = open(path.join(BASE_DIR, "names.txt"))
6
7   data = file.read()
8   print(data)
9
10  data = file.readline()
11  print(data)
12
13  data = file.readlines()
14  print(data)
15
```

```
1   import os
2   from os import path
3
4   BASE_DIR = path.join(path.dirname(__file__), "files")
5   file = open(path.join(BASE_DIR, "names.txt"))
6
7   linhas = file.readlines()
8   for linha in linhas:
9       print(linha.strip())
10      # strip remove os caracteres especiais
11      # da frente e do fim da string
12
```

with Statement

EDUCAR PARA TRANSFORMAR

- "with statement" em Python é usado no tratamento de exceções para tornar o código mais limpo e muito mais legível.
- Simplifica o gerenciamento de recursos comuns, como fluxos de arquivos.

```
15  # 1) without using with statement
16  file = open("file_path", "w")
17  file.write("hello world !")
18  file.close()
19
20  # 2) without using with statement
21  file = open("file_path", "w")
22  try:
23      file.write("hello world")
24  finally:
25      file.close()
26
27  # 3) using with statement
28  with open("file_path", "w") as file:
29      file.write("hello world !")
30
```


hands
on

Gravando um arquivo de texto

EDUCAR PARA TRANSFORMAR

- A gravação de uma linha em uma arquivo é feita de maneira simples com o uso da função write("texto").
- É possível escrever diversas linhas em um arquivo com o auxílio de um laço de repetição.

  - write() – escreve uma linha do arquivo
  - writelines() – escreve um conjunto de linhas no arquivo

```python
import os
from os import path

BASE_DIR = path.join(path.dirname(__file__), "files")
names = ["joao", "maria", "pedro", "andre", "jose"]

with open(path.join(BASE_DIR, "names.txt"), "w") as file:
    for name in names:
        file.write(name)
        file.write("\n")

```

```python
import os
from os import path

BASE_DIR = path.join(path.dirname(__file__), "files")
# names = ["joao", "maria", "pedro", "andre", "jose"]
# names = ["joao\n", "maria\n", "pedro\n", "andre\n", "jose\n"]
names = ("joao\n", "maria\n", "pedro\n", "andre\n", "jose\n")

# with open(path.join(BASE_DIR, "names.txt"), "w") as file:
#     for name in names:
#         file.write(name)
#         file.write("\n")

with open(path.join(BASE_DIR, "names.txt"), "w") as file:
    file.writelines(names)

```


hands
on

Recebendo arquivos como parâmetros

EDUCAR PARA TRANSFORMAR

- Uma funcionalidade importante é a capacidade de receber, na chamada de um script, parâmetros.
- Esses podem ser desde dados utilizados durante a execução do programa até mesmo nomes de arquivos que serão processados ou criados.
    - python meu_script.py meu_arquivo.txt
- Nesse exemplo, o script meu_script.py deve conter a importação do módulo sys, o qual permite receber valores passados na chamada de um programa em Python.

```
1  import sys
2
3  for arg in sys.argv:
4      print("parametro ", arg)
5
```


hands
on

Obrigado!

EDUCAR PARA TRANSFORMAR

Prof. Henrique Mota

profhenriquemota@gmail.com