Aula 10 - Manipução de Arquivos

June 10, 2021

1 Manipulação de arquivos

1.1 Caminhos de abertura

- Caminhos Relativos => A partir da pasta atual
  - Ex: arquivos//arquivo.txt
- Arquivos na pasta atual => Apenas o nome do arquivo
  - Ex: arquivo.txt
- Caminho Total => Todo o caminho
  - Ex: C://caminho//arquivo.txt

1.2 Permissões de abertura

- 'r' = Read => Apenas escrita (Permissão padrão do Python)
- 'a' = Append => Ler e Escrever sem deletar o conteúdo (adicionar conteúdo)
- 'w' = Write => Escrever no arquivo a partir do zero (deleta o conteúdo)
- 'w+' = Write+ => Escrever no arquivo a partir do zero (deletando o conteúdo) e criando caso ele não exista

1.3 Tipos de Leitura

- 't' = Text => Arquivo de texto
- 'b' = Binary => Arquivo binários (como imagens)

[4]:
```
f = open("arquivos//arquivo.txt","r") # Read => Apenas Escrita
print(f)
```

```
<_io.TextIOWrapper name='arquivos//arquivo.txt' mode='r' encoding='cp1252'>
```

[5]:
```
f = open("arquivos//arquivo.txt","a") # Append => Leio e quero adicionar textos
print(f)
```

```
<_io.TextIOWrapper name='arquivos//arquivo.txt' mode='a' encoding='cp1252'>
```

[6]:
```
f = open("arquivos//arquivo.txt","w") # Write => Escrita => Quero escrever␣
 ↪nesse arquivo a partir do zero
print(f)
```

```
<_io.TextIOWrapper name='arquivos//arquivo.txt' mode='w' encoding='cp1252'>
```

[8]:
```
f = open("arquivos//arquivo2.txt", "w+") # Arquivo inexistente
print(f)
```

```
<_io.TextIOWrapper name='arquivos//arquivo2.txt' mode='w+' encoding='cp1252'>
```

[10]:
```
f = open("arquivos//arquivo3.txt", "r") # Arquivo inexistente
print(f)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-10-d010f47133c4> in <module>
----> 1 f = open("arquivos//arquivo3.txt", "r") # Arquivo inexistente
      2 print(f)

FileNotFoundError: [Errno 2] No such file or directory: 'arquivos//arquivo3.txt
```

[11]:
```
f = open("arquivos//arquivo.txt") # Read => Apenas Escrita
print(f)
```

```
<_io.TextIOWrapper name='arquivos//arquivo.txt' mode='r' encoding='cp1252'>
```

[13]:
```
f = open("arquivos//arquivo.txt", "rt") # T => Texto => TextIOWrapper
print(f)
```

```
<_io.TextIOWrapper name='arquivos//arquivo.txt' mode='rt' encoding='cp1252'>
```

[14]:
```
f = open("arquivos//arquivo.txt", "rb") # B => Binário => BufferedReader
print(f)
```

```
<_io.BufferedReader name='arquivos//arquivo.txt'>
```

1.4 Abertura e Fechamento de Arquivo

Quando eu abro um arquivo eu estou dizendo que eu estou usando ele e ele deve permanecer existente até o final do uso.
Quando eu termino de usar eu preciso informar isso através da função **close()**.

[53]:
```
f = open("arquivos//arquivo3.txt") # Abrir o arquivo
```

[54]:
```
f.close() # Fechei o arquivo
```

1.5 Leitura do arquivo

[15]:
```
f = open("arquivos//arquivo.txt") # Read => Apenas Escrita
print(f)
```

```
<_io.TextIOWrapper name='arquivos//arquivo.txt' mode='r' encoding='cp1252'>
```

Para ler todo o arquivo em uma string só usamos o **read()**

[16]:
```
print(f.read()) # Read => Lê todo o arquivo em uma string só
```

```
primeira linha
segunda linha
terceira linha
quarta linha
quinta linha
```

Para ler apenas uma linha usamos o **readline()**

[21]:
```
f = open("arquivos//arquivo.txt")
f.readline()
```

[21]:
```
'primeira linha\n'
```

A cada execução o Python lê a próxima linha, ou seja, 'ignora' a leitura das anteriores.
Em outras palavras ele 'segue a diante'

[27]:
```
f.readline()
```

[27]:
```
''
```

Para ler todas as linhas uma por uma e retornar em uma lista usamos o **readlines()**

[30]:
```
f = open("arquivos//arquivo.txt")
linhas = f.readlines()
print(linhas)
```

```
['primeira linha\n', 'segunda linha\n', 'terceira linha\n', 'quarta linha\n',
'quinta linha']
```

[31]:
```
linhas[0]
```

[31]:
```
'primeira linha\n'
```

[33]:
```
type(linhas)
```

[33]:
```
list
```

[35]:
```
linhas = f.readlines()
linhas
```

[35]:
```
[]
```

Repare que ao chamarmos pela segunda vez a função retornar uma lista vazia pois ela, na verdade, **retorna as linhas restantes.**
Como, ao abrir o arquivo, restavam todas as linhas então ela retornou todas as linhas.

Confundiu? Veja se este exemplo clareia as coisas.

[42]:
```
f = open("arquivos//arquivo.txt")
primeira_linha = f.readline() # Leio o inicio do arquivo
segunda_linha = f.readline() # Move o cabeçote de leitura para frente indo para␣
 ↪o 'novo' (ainda não lido) => Lê 2ª linha
proximas_linhas = f.readlines() # Lê as linhas restantes
```

[44]:
```
print("Primeira Linha:",primeira_linha)
print("Segunda Linha:",segunda_linha)
print("Próximas Linhas:",proximas_linhas)
```

```
Primeira Linha: primeira linha

Segunda Linha: segunda linha

Próximas Linhas: ['terceira linha\n', 'quarta linha\n', 'quinta linha']
```

Para ler apenas **X** caracteres uso a função **Read()** passando a quantidade por parâmetro

[48]:
```
f = open("arquivos//arquivo.txt")
print(f.read(10)) # Passo por parâmetro a quantidade que quero ler
```

```
primeira l
```

Podemos percorrer linha por linha através de um **loop for**

[55]:
```
f = open("arquivos//arquivo.txt")
for l in f:
    print(l)
```

```
primeira linha

segunda linha

terceira linha

quarta linha

quinta linha
```

1.6 Escrevendo em arquivos

Lembrete: - 'w' => Write => Escrever a partir do zero => Deleto tudo o que existe - 'a' => Append => Incremento o conteúdo => Adiciono ao que já existe

[65]:
```
f = open("arquivos//arquivo2.txt", "w")
f.write("Apaguei todo o conteúdo e escrevi este aqui :)")
f.close()
```

[68]:
```
f = open("arquivos//arquivo2.txt", "r")
print(f.read())
```

```
f.close()
```

```
Apaguei todo o conteúdo e escrevi este aqui :)
Estou adicionando também esse conteúdo :)
```

[67]:
```
f = open("arquivos//arquivo2.txt", "a")
f.write("\nEstou adicionando também esse conteúdo :)")
f.close()
```

[ ]:
```

```