Python - Ler arquivos

Para ler programaticamente os dados de um arquivo usando Python, ele deve ser aberto primeiro. Use a função open() integrada -

```
file object = open(file_name [, access_mode][, buffering])
```

Aqui estão os detalhes dos parâmetros -

- **file_name** - O argumento file_name é um valor de string que contém o nome do arquivo que você deseja acessar.
- **access_mode** - O access_mode determina o modo em que o arquivo deve ser aberto, ou seja, leitura, gravação, acréscimo, etc. Este é um parâmetro opcional e o modo de acesso ao arquivo padrão é leitura (r).

Estas duas declarações são idênticas -

```
fo = open("foo.txt", "r")
fo = open("foo.txt")
```

Para ler os dados do arquivo aberto, use o método read() do objeto Arquivo. É importante observar que strings Python podem ter dados binários além dos dados de texto.

Sintaxe

```
fileObject.read([count])
```

Parâmetros

- **count** - Número de bytes a serem lidos.

Aqui, o parâmetro passado é o número de bytes a serem lidos do arquivo aberto. Este método inicia a leitura desde o início do arquivo e se faltar contagem, ele tenta ler o máximo possível, talvez até o final do arquivo.

Exemplo

```
# Open a file
fo = open("foo.txt", "r")
```

```
text = fo.read()
print (text)

# Close the opened file
fo.close()
```

Ele produzirá a seguinte **saída** -

```
Python is a great language.
Yeah its great!!
```

Leitura em modo binário

Por padrão, as operações de leitura/gravação em um objeto de arquivo são executadas em dados de string de texto. Se quisermos lidar com arquivos de diferentes tipos, como mídia (mp3), executáveis (exe), imagens (jpg) etc., precisamos adicionar o prefixo 'b' ao modo de leitura/gravação.

Supondo que o arquivo **test.bin** já tenha sido escrito em modo binário.

```
f=open('test.bin', 'wb')
data=b"Hello World"
f.write(data)
f.close()
```

Precisamos usar o modo 'rb' para ler o arquivo binário. O valor retornado do método read() é decodificado primeiro antes da impressão

```
f=open('test.bin', 'rb')
data=f.read()
print (data.decode(encoding='utf-8'))
```

Ele produzirá a seguinte **saída** -

```
Hello World
```

Ler dados inteiros do arquivo

Para escrever dados inteiros em um arquivo binário, o objeto inteiro deve ser convertido em bytes pelo método to_bytes().

```
n=25
n.to_bytes(8,'big')
```

```
f=open('test.bin', 'wb')
data=n.to_bytes(8,'big')
f.write(data)
```

Para ler de um arquivo binário, converta a saída da função read() em inteiro usando a função from_bytes().

```
f=open('test.bin', 'rb')
data=f.read()
n=int.from_bytes(data, 'big')
print (n)
```

Ler dados flutuantes do arquivo

Para dados de ponto flutuante, precisamos usar o módulo **struct** da biblioteca padrão do Python.

```
import struct
x=23.50
data=struct.pack('f',x)
f=open('test.bin', 'wb')
f.write(data)
```

Descompactando a string da função read() para recuperar os dados flutuantes do arquivo binário.

```
f=open('test.bin', 'rb')
data=f.read()
x=struct.unpack('f', data)
print (x)
```

Usando o modo r+

Quando um arquivo é aberto para leitura (com 'r' ou 'rb'), não é possível gravar dados nele. Precisamos fechar o arquivo antes de realizar outra operação. Para realizar as duas operações simultaneamente, temos que adicionar o caractere '+' no parâmetro mode. Portanto, o modo 'w+' ou 'r+' permite o uso dos métodos write() e read() sem fechar um arquivo.

O objeto File também suporta a função seek() para retroceder o fluxo para ler a partir de qualquer posição de byte desejada.

A seguir está a sintaxe do método seek() -

```
fileObject.seek(offset[, whence])
```

Parâmetros

- **offset** - Esta é a posição do ponteiro de leitura/gravação no arquivo.
- **wherece** - Isto é opcional e o padrão é 0, o que significa posicionamento absoluto do arquivo, outros valores são 1, o que significa busca relativa à posição atual e 2 significa busca relativa ao final do arquivo.

Vamos usar o método seek() para mostrar como ler dados de uma determinada posição de byte.

Exemplo

Este programa abre o arquivo no modo w+ (que é um modo de leitura e gravação), adiciona alguns dados. Ele busca uma determinada posição no arquivo e substitui seu conteúdo anterior por um novo texto.

```
fo=open("foo.txt","r+")
fo.seek(10,0)
data=fo.read(3)
print (data)
fo.close()
```

Ele produzirá a seguinte **saída** -

```
rat
```


Leitura/gravação simultânea em Python

Quando um arquivo é aberto para escrita (com 'w' ou 'a'), não é possível lê-lo e vice-versa. Fazer isso gera o erro UnSupportedOperation. Precisamos fechar o arquivo antes de realizar outra operação.

Para realizar as duas operações simultaneamente, temos que adicionar o caractere '+' no parâmetro mode. Portanto, o modo 'w+' ou 'r+' permite o uso dos métodos write() e read() sem fechar um arquivo. O objeto File também suporta a função seek() para retroceder o fluxo para qualquer posição de byte desejada.

O método search()

O método seek() define a posição atual do arquivo no deslocamento. O argumento wherece é opcional e o padrão é 0, o que significa posicionamento absoluto do arquivo, outros valores são 1, que significa busca relativa à posição atual e 2 significa busca relativa ao final do arquivo.

Não há reembolso. Observe que se o arquivo for aberto para anexar usando 'a' ou 'a+', qualquer operação seek() será desfeita na próxima gravação.

Se o arquivo for aberto apenas para gravação no modo anexar usando 'a', este método é essencialmente autônomo, mas permanece útil para arquivos abertos no modo anexar com leitura habilitada (modo 'a+').

Se o arquivo for aberto em modo texto usando 't', apenas os deslocamentos retornados por Tell() serão válidos. O uso de outros deslocamentos causa comportamento indefinido.

Observe que nem todos os objetos de arquivo são pesquisáveis.

Sintaxe

A seguir está a sintaxe do método seek() -

```
fileObject.seek(offset[, whence])
```

Parâmetros

- **offset** - Esta é a posição do ponteiro de leitura/gravação no arquivo.
- **wherece** - Isto é opcional e o padrão é 0, o que significa posicionamento absoluto do arquivo, outros valores são 1, o que significa busca relativa à posição atual e 2 significa busca relativa ao final do arquivo.

Vamos usar o método seek() para mostrar como a operação simultânea de leitura/gravação em um arquivo pode ser feita.

O programa a seguir abre o arquivo no modo w+ (que é um modo de leitura e gravação) e adiciona alguns dados. Ele busca uma determinada posição no arquivo e substitui seu conteúdo anterior por um novo texto.

Exemplo

```
# Open a file in read-write mode
fo=open("foo.txt","w+")
fo.write("This is a rat race")
fo.seek(10,0)
data=fo.read(3)
fo.seek(10,0)
fo.write('cat')
fo.seek(0,0)
data=fo.read()
print (data)
fo.close()
```

Saída

```
This is a cat race
```