

Leitura dados pré-existentis

Na sua vida profissional, será comum você ser chamado a processar dados gerados em outro local, por outras pessoas e em outros ambientes operacionais. Com a constante digitalização dos conteúdos e das informações, tal demanda já existe hoje e tende a se tornar mandatória com o passar dos anos.

Tipo de arquivo A primeira questão que se precisa definir é com relação ao tipo do arquivo a manusear. Fundamentalmente existem 2 tipos, que são conhecidos como BINÁRIOS e de TEXTO. A distinção é simples, os arquivos binários tem uma lógica de construção e funcionamento que dependem do ambiente gerador. Não seguem algum padrão, nem têm operação automática. Uma maneira fácil de identificar este tipo de arquivo é dar um comando de MOSTRAR o conteúdo. Neste tipo de arquivo o conteúdo será composto de caracteres ininteligíveis, não sendo possível intuir nada. (Em Windows/DOS o comando de mostrar é TYPE. Em Linux, o comando é CAT). Outro jeito fácil é abrir o arquivo para edição usando um editor pleno (o notepad, o N++, o scintilla, ou qualquer programa similar). Tendo aberto um arquivo BINÁRIO seu conteúdo será impossível de entender.

Por não serem padrão, os arquivos binários não serão trabalhados aqui. Quando tiver que usá-los, você vai precisar pedir ajuda ao suporte técnico, ou então à pessoa ou organização que gerou o arquivo. Os arquivos tipo TEXTO, como o nome diz, contém apenas caracteres visíveis/legíveis. Neste tipo de arquivo, os dados estão organizados em linhas, e cada linha é encerrada por dois caracteres padrão conhecidos como CR (carriage return) e LF (line feed). Estes 2 caracteres não são visíveis, mas é possível intuir sua presença pelo comportamento dos dados enquanto são mostrados.

Uma maneira segura de "olhar" dentro de qualquer arquivo seja BINÁRIO seja de TEXTO é através de um editor binário, como por exemplo o HEXEDIT (freeware, procure na Internet).

Modo de acessar Há 2 maneiras básicas de acessar um arquivo de entrada. A primeira usa uma propriedade dos sistemas operacionais, chamada **redirecionamento**. Suponha que seu programa de nome ALFA33, lê dados do teclado (através de readline()). Se, ao chamar o programa, você digitar ALFA33 < ARQENTR tudo vai funcionar como se o conteúdo do arquivo ARQENTR fosse digitado pelo teclado como resposta aos comandos scanf do programa.

Ao agir assim, o programa fonte não sofre nenhuma alteração e a vantagem é que o mesmo programa, sem nenhuma modificação, aceita dados tanto do teclado quanto de um arquivo.

Embora não vá ser tratado aqui, o redirecionamento também pode ser feito para a saída, e daí, os comandos printf ao invés de imprimir no vídeo, vão gerar um arquivo em disco. Para redirecionar a saída, fazer ALFA33 > ARQSAIDA.

Os dois redirecionamentos podem ser usados juntos: ALFA33 < ARQENTR > ARQSAIDA, e obviamente o redirecionamento só pode ser usado para arquivo TEXTO.

A outra maneira de acessar arquivos em disco, é declarar isto explicitamente, como se verá a seguir:

Open/Close: Dentro de qualquer ambiente computacional, um arquivo é conhecido pelo seu nome (e eventualmente, este nome é composto, incluindo o caminho até o mesmo). Por exemplo, um arquivo pode se chamar c:/dados/versao2/cad-pessoas-set-12.txt. Neste exemplo o arquivo é cad-pessoas-set-12.txt que reside no disco que está em C: e dentro dele, no diretório dados e sub-diretório versao2.

Já dentro dos programas Python, existe um nome interno, que é sempre o mesmo. (Esta estratégia é inteligente, já que não teria sentido ter que reescrever o programa cada vez que o nome do arquivo de dados fosse modificado).

A associação entre nome externo (no disco) e interno (no programa) é feita por uma operação

chamada OPEN e a dissociação destes dois elementos é feita pela operação CLOSE ou pelo encerramento do programa, o que ocorrer primeiro.

Neste exercício, você vai receber diversos arquivos, todos do tipo TEXTO, cada um deles com um determinado formato e para cada um, você deve calcular o que é pedido. Antes de começar, estude o arquivo recebido e tente entender qual é a lógica de sua construção. Use o HEXEDIT ou similar para fazer isso. Veja também o arquivo (se ele for do tipo TEXTO) com um editor comum, como o SCINTILLA ou mesmo o NOTEPAD. Descubra o tamanho dos registros e a quantidade de registros que existem no arquivo. Considere neste caso o conceito de registro como quase sinônimo de linha.

Leitura As leituras de dados (uma para cada registro) são feitas usando-se o comando **readline** do Python (entre outras maneiras). Ele entrega uma linha lida no arquivo no formato string.

Um exemplo completo Suponha um arquivo contendo em cada linha, um valor real referente a dinheiro e um número inteiro referente a um certo número de dias. Os códigos CR+LF não são visíveis, mas estão lá:

2500.00 56 CR+LF
1270.77 66 CR+LF
...

No Python, devem ser executados pelo menos estes comandos:

a) ref = open("c:/lixo/bla bla", "r")
b) lin = ref.readline()
c) lis = lin.split()
d) campo1 = int(lis[0]) ...
d) ref.close()

Observações:

1. O nome externo é c:/lixo/bla bla. Note que as barras são viradas à direita, por exigência do Python.
2. O dado vai ser aberto para leitura (método r, de READ).
3. O nome interno é ref.
4. readline pode estar dentro de um loop, para ler todos os registros. Neste caso, a saída será quando o tamanho de lin for igual a zero.
5. split transforma uma string em uma lista. O separador de elementos é o espaço em branco.
6. Finalmente, o comando close deve ser emitido uma única vez ao final do processamento.

Acompanhe agora o programa Python completo

```
def programa():  
    ref=open("c:/lixo/oba.txt","r")  
    lin = ref.readline()  
    while 0!=len(lin):  
        lis=lin.split()  
        a = int(lis[0])  
        b = float(lis[1])  
        ...  
        lin = ref.readline()  
    ref.close()
```

Para testar sua lógica: Se quiser testar a lógica de seus programas, processe os arquivos de nomes EXEF00E1, EXEF00E2 e EXEF00E3. Para eles os 10 valores calculados serão:

3017.39 5498 386
258648.89 249233.21 114739.33
17.7 184 2 128

Para você fazer

Exercício 1: Você deve pegar o arquivo de nome

XXXF01E1

e estudá-lo. Ele é formado por 1000 números inteiros, cada um representando o rendimento mensal de um determinado empregado de uma certa empresa em reais (sem o uso de centavos, por isso o tipo inteiro). Depois de ler os dados e criar um vetor, deve:

Informe o valor médio do rendimento	1.
O maior rendimento	2.
Quantas pessoas ganharam mais do que 20% do rendimento médio	3.

Exercício 2: Agora, o arquivo a ler é

XXXF01E2

nele estão 500 duplas de dados que se referem a dívidas que pessoas/empresas têm com você. O primeiro valor é um número real significando o valor da dívida em moeda nacional. O segundo valor é um inteiro, indicando o número de dias de atraso. Leia os dados e construa uma matriz de 500 x 2 valores. Depois, responda:

Valor total da dívida	4.
Valor total da dívida se forem desconsideradas as dívidas inferiores a R\$ 200,00	5.
Valor total da dívida se forem desconsideradas as dívidas com prazo de atraso maior do que 89 dias	6.

Exercício 3: Neste exercício você deve ler o arquivo

XXXF01E3

que contém uma lista de 365 conjunto de valores. Cada um deles descreve as condições meteorológicas de Curitiba, no ano de 2016. Para cada dia, os dados informam:

- Temperatura máxima no dia, em graus centígrados com 1 decimal.
- Temperatura mínima no dia, idem acima.
- Umidade do ar ao meio dia, em percentagem com 1 decimal.
- Precipitação pluvial, em mm como um número inteiro.
- Horas de insolação, em horas inteiras.

Processe os dados e informe:

Temperatura média no ano: (max+min)/2 a cada dia...	7.
Quantos dias tiveram mais de 3h de insolação	8.
Qual o semestre mais chuvoso (1sem=182; 2sem=183)	9.
Quantos dias tiveram umidade abaixo de 30%	10.