Arquivos em Python

Um arquivo é uma coleção de dados organizados. Um arquivo é gravado em alguma mídia magnética, ótica ou física. Esta pode ser um CD, um pendrive ou um disco magnético, por exemplo. No passado podia ser um deck de cartões de papel perfurados. Pode também ser armazenado na núvem, substantivo que descreve algum servidor em algum lugar no hiperespaço da Internet.

Um arquivo não está associado a alguma linguagem, já que uma vez gravado e salvo pode ser processado por qualquer programa escrito em qualquer linguagem. Nesta folha, vamos estudar e exercitar os comandos Python que manuseiam arquivos.

Porque usar arquivos

- O arquivo é permanente. Ou seja, mesmo que você desligue o computador onde o arquivo foi gerado ele permanece existindo mesmo após a conclusão da sessão onde foi gerado.
- O arquivo pode ser transferido, cedido ou recebido. De fato, arquivos são a maneira pela qual as pessoas e organizações trocam informações.
- O arquivo oferece segurança contra perda. Essa é a razão pela qual fazemos $back-ups$, que nada mais são do que arquivos cópia das coisas que nos interessam e que não queremos perder por acidente ou distração.
- O arquivo contém bens. Nessa sociedade digitalizada, muitos bens estão armazenados em arquivos. Livros, filmes, músicas, documentos e muitas coisas mais estão guardadas em arquivos PDF, MP3, etc.
- Arquivos oferecem identificação. Uma característica importante dos nossos dias é a certificação digital, que nada mais é que um arquivo cuja posse e manuseio certifica que seu proprietário é quem diz ser, além de qualquer dúvida razoável.
- Podem ser dinheiro. A propriedade de bitcoins e outras moedas virtuais se dá apenas pela posse do conhecimento de um número de 70 dígitos. Tradicionalmente, mas não obrigatoriamente este conhecimento se dá pela posse de um arquivo que o tem gravado.

Os arquivos são gerenciados pelo sistema operacional do computador que se está usando de maneira que toda a complexidade associada às operações com o arquivos ficam escondidas pelo sistema operacional (e pelo Python nesta folha).

Cada arquivo possui um nome externo (pelo qual o mundo exterior o conhece como por exemplo: EGUINHAPOCOTO.MP3 ou NOTASFINAIS202-2SEM18.PDF e um nome interno pelo qual o programa que o manuseia o conhece.

Age-se assim para permitir que o mesmo programa possa processar muitos arquivos sempre da mesma maneira e sem precisar modificar o programa quando novos arquivos tiverem que ser processados.

Quem faz a ligação entre nome externo e nome interno é uma operação muito frequente em programação e muito conhecida dos programadores denominada ABERTURA do arquivo.

Na abertura oferecem-se os dois nomes e o Python ao processá-la busca o arquivo no substrato físico (e eventualmente no diretório e subdiretórios) onde ele deve estar e aloca áreas de memória e algoritmos necessários para manuseá-lo. Se o arquivo não for encontrado onde deveria estar o Python sinaliza este fato oferecendo um erro específico.

Em python, abre-se um arquivo com a função `open` cujo formato é:

```
nome_interno=open("nome_externo","modalidade")
```

O `nome_interno` é a identificação do arquivo dentro do programa Python. O `nome_externo` é o nome real, físico do arquivo e também o caminho completo (`drive: diretório\subdiretório[...]\nome.tipo` do arquivo. Já a `modalidade` é um ou mais caracteres que informam o que se fará com o arquivo. Os valores são:

`r`	o arquivo será lido
`w`	o arquivo será gravado começando zerado
`a`	gravação, mas preservando o que já há.
`b`	arquivo em modo binário
`+`	atualização: leitura e escrita juntas

Os modos podem ser combinados (por exemplo: `rb`). A função `open` retorna um objeto do tipo `file` (arquivo). É este objeto que vai permitir manipular o arquivo. Depois de aberto, o arquivo vai sofrer operações de `read` para leitura e `write` para gravação. Finalmente, depois que o processamento acabar o arquivo deve sofrer uma operação de `close` que vem a ser o encerramento do ciclo. É importante fechar o arquivo ao final, pelo menos por 2 razões:

- Os recursos alocados ao arquivo sao liberados e
- Os dados do último bloco, que está em memória, são garantidamente gravados em disco

A operação de close, muitas vezes é ignorada, pois ao encerrar o programa o Python (na verdade o S.O.) fecha todos os arquivos.

Uma primeira experiência: Digite e execute o seguinte programa Python

```
arq = open("arquivonums.txt","w")
for lin in range(1,101):
   arq.write("%d - gravando dados \n" % lin)
arq.close()
```

Se der tudo certo, não vai aparecer nada na tela. Mas, se você carregar o Windows explorer e for no diretório em que está trabalhando, verá o arquivo lá. Se clicar 2 vezes nele, o Notepad vai abrir o arquivo. Pronto, você acaba de criar um arquivo com um programa e abrí-lo com outro.

O programa para ler o arquivo gravado pode ser

```
arq=open("arquivonums.txt","r")
for lin in arq.readlines():
   print(lin)
arq.close()
```

HTML

Uma página WEB nada mais é do que um arquivo. Você pode criá-la usando um editor (como o notepad) ou pode criar um programa Python que grava uma página WEB. Tais arquivos são chamados texto plano, o que significa que não contém nenhum caractere não imprimível. Logo, podem ser criados via editor de textos comum.

Nosso arquivo acima (`arquivonums.txt`) já pode ser visto num browser, embora ainda não tenha marcações HTML. Para fazer isto, localize o arquivo com o windows explorer e arraste-o para uma sessão já aberta com o browser.

O padrão para escrever arquivos ricos (bonitos) na web é chamado HTML (Hyper Text Markup Language) genial criação de um físico do CERN, Tim Berners Lee. Para deixar a coisa bonita, usam-se tags, que vem a ser instruções de marcação. Cada tag está escrita entre parêntes bicudos e há uma marcação de ínicio e outra de final. Esta última é igual à primeira, a menos de uma barra. Veja por exemplo, a marcação `<body>` que sinaliza o corpo do documento:

```
<body> Este é um documento HTML </body>
```

Existem centenas de tags, é um curso à parte. Vamos estudar só umas poucas para poder brincar com os browsers. Eis uma lista bem parcial

`html`	inicia um arquivo html
`title`	dá o título ao arquivo html
`body`	define o corpo do arquivo html
`h1, h2, h3...`	níveis de cabeçalhos
`a href`	hiper link

Agora, vamos juntar as 2 coisas: gravar um arquivo em padrão HTML. Copie e execute o seguinte programa Python:

```
pag=open("arq.html","w",encoding='utf-8')
pag.write("<html>")
pag.write("<title>Nossa primeira página</title>")
pag.write("<body><h1>Veja o que fazer...</h1>")
pag.write("<ul>")
for i in range(1,11):
   pag.write("<li>  item número  %d  "%i)
pag.write("</ul> </body></html>")
pag.close()
```

Rode e imagine tudo o que dá para fazer aqui...

Para ler o arquivo

```
import numpy as np
def f933(nome):
  lst=[]
  respo=[]
  a=open(nome,"r")
  for linha in a.readlines():
    lin=linha.split()
    for j in range(0,10):
      lst.append(int(lin[j]))
  mai=-99999
  men=+99999
  som=0
  for i in range(0,1000):
    som=som+lst[i]
    if lst[i]>mai:
      mai=lst[i]
    if lst[i]<men:
      men=lst[i]
  respo.append(mai-men)
  med = som/1000
  respo.append(med)
  som=0
  for i in range(0,1000):
    som=som+((lst[i]-med)**2)
  som=som/1000
  respo.append(som**0.5)
  lstord=np.sort(lst)
  respo.append((lstord[499]+lstord[500])/2)
  dic={lst[1]:1}
  for i in range(1,1000):
    a=dic.get(lst[i])
    if a is None:
      dic[lst[i]]=1
    else:
      dic[lst[i]]=dic[lst[i]]+1
  mai=-9999999
  for ch,da in dic.items():
    if da>mai:
      mai=da
      queme=ch
  respo.append(queme)
  return respo
print(f933("caminho e nome"))
```

☞ Para você fazer

No arquivo

`F9330001.myd`

que deve estar publicado em www.pkantek.com.br, você vai encontrar 1000 números que configuram um distribuição qualquer. Leia o arquivo, processe os dados e ao final descubra:

1. A amplitude dos dados
2. A média dos valores
3. O desvio padrão da distribuição
4. A mediana (o valor do meio)
5. A moda (o valor mais frequente)

Existe no mesmo local o arquivo `f933exe1.myd`. Ao processá-lo e se o seu programa estiver funcionando corretamente, ele deve apresentar os seguintes valores [1997, 995.7, 582.0, 1004.0, 1051]

Responda aqui, sempre que o número não for inteiro, com 1 casa decimal:

1	2	3	4	5