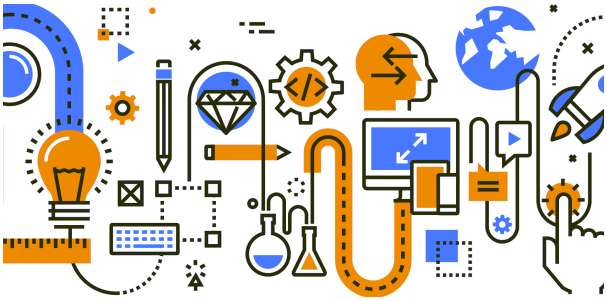
}

MC102 - Algoritmos e Programação de Computadores



Análise de dados, Arquivos CSV e Química!

Uma das tarefas em que a Computação é mais utilizada atualmente é a análise de dados. Este tipo de atividade pode ter muitas formas diferentes — processar dados obtidos por meio de questionários, encontrar padrões em informações capturadas por sensores como termômetros e velocímetros, analisar dados de experimentos em laboratório e muito mais.

Arquivos CSV

Você certamente já está acostumado com dados organizados em tabelas ou planilhas, como no exemplo abaixo:

Elemento Químico	Número atômico
Hidrogênio	1
Hélio	2
Lítio	3
Berílio	4
Boro	5
Carbono	6

Um arquivo CSV é uma forma compacta de representar dados em tabelas. CSV são as inicias em inglês de Comma-Separated Values, que significa "valores separados por vírgulas". Como o nome sugere, teremos um arquivo texto com vírgulas separando os valores das colunas.

```
Elemento Químico, Número Atômico
Hidrogênio, 1
Hélio, 2
Lítio, 3
Berílio, 4
Boro, 5
Carbono, 6
```

A existência do cabeçalho com nomes das colunas na primeira linha é opcional. O formato CSV tem diversas vantagens. Ele é aceito por diversos softwares diferentes, possui uma estrutura simples e pode ser editado ou visualizado usando até mesmo um editor de texto simples.

Objetivo desta tarefa

Nesta tarefa, iremos fazer uma análise simples dos dados obtidos por colegas da Engenharia Química numa disciplina de laboratório. Caso você queira se aprofundar no assunto, procure o(a) aluno(a) de Engenharia Química mais próximo(a) e pergunte sobre a disciplina EQ801 - Laboratório de Engenharia Química III. Nós iremos trabalhar com dados no formato abaixo, em que as medidas de temperatura estão expressas em Kelvin e as de pressão em kPa:

Temperatura	Pressão Medida	Pressão Esperada	ln(Pressão Esperada)
326.75	39.9	39.92	3.69
333.05	49.75	49.76	3.91
338.45	59.89	59.72	4.09
340.95	64.90	64.86	4.17

A teoria por trás desse experimento está baseada na Equação de Clausius-Clapeyron que correlacionou as três principais variáveis consideradas no estudo dos gases (pressão, volume e temperatura) e o número de mols. Em particular, trabalharemos como uma simplificação desta equação que permite o cálculo de pressão de vapor em função da temperatura.

$$\ln(P^{VAP}) = A + B/Temperatura$$

Note que nós precisaremos do valor de duas constantes: `A` e `B`. Estas constantes são ajustadas a dados experimentais de pressão de vapor de uma substância. Quanto **menor**, em valores absolutos, for a diferença entre o valor obtido por essa fórmula e o logaritmo natural da pressão esperada, melhor é a aproximação de A e B! Veja os exemplos na tabela abaixo:

Temperatura	A	B	A + B/Temperatura	ln(Pressão Esperada)	Diferença
326.75	14	-3800	14 - 3800/326.75 = 2,37	3.69	1.32
326.75	15	-3805	15 - 3805/326.75 = 3.36	3.69	0.33

Nós iremos trabalhar com constantes `A` e `B` distintas e aplicar esta fórmula para os dados obtidos num experimento, comparando-os aos valores esperados. Para isso, no entanto, precisamos aprender a processar arquivos.

Abrindo e Processando Arquivos

Nós podemos abrir e verificar o conteúdo de um arquivo usando a função `open()` do Python. Suponha que você possua em seu diretório local um arquivo chamado evidencias.txt que contém o início da letra da tradicional música "Evidências", de Chitãozinho e Xororó:

```
E nessa loucura de dizer que não te quero
Vou negando as aparências
Disfarçando as evidências
Mas pra que viver fingindo
Se eu não posso enganar meu coração
Eu sei que te amo
Chega de mentiras
De negar o meu desejo
Eu te quero mais que tudo
Eu preciso do seu beijo
Eu entrego a minha vida
Pra você fazer o que quiser de mim
Só quero ouvir você dizer que sim

Diz que é verdade, que tem saudade
Que ainda você pensa muito em mim
Diz que é verdade, que tem saudade
Que ainda você quer viver pra mim
```

Nós podemos abrir esse arquivo com o comando `open("evidencias.txt")`. Esse comando por si só nos dará um iterador pelo arquivo, portanto podemos utilizá-lo dentro de um comando `for` para passear por todas as linhas:

```
for linha in open("evidencias.txt"):
    print(linha, end="")

E nessa loucura de dizer que não te quero
Vou negando as aparências
Disfarçando as evidências
Mas pra que viver fingindo
Se eu não posso enganar meu coração
(...)
```

Observe que toda linha lida termina por um caractere `\n`. Desta forma, para não haver linhas em branco extras na saída, alteramos o finalizador padrão do `print()` para `end=""`.

Note que a variável `linha` guardará dentro de si uma `string` comum e, portanto, podemos tudo o que é possível fazer com uma `string` como, por exemplo, colocar todas as letras em maiúsculo:

```
for line in open("evidencias.txt"):
    print(linha.upper(), end="")

E NESSA LOUCURA DE DIZER QUE NÃO TE QUERO
VOU NEGANDO AS APARÊNCIAS
DISFARÇANDO AS EVIDÊNCIAS
MAS PRA QUE VIVER FINGINDO
SE EU NÃO POSSO ENGANAR MEU CORAÇÃO
(...)
```

Outros métodos interessantes que podemos aplicar a um arquivo aberto são `open().read()`, que retorna todo o conteúdo do arquivo numa única string e `open().readlines()`, que retorna uma lista com uma `string` por linha:

```
conteudo = open("evidencias.txt").read()
print(conteudo)

'E nessa loucura de dizer que não te quero\nVou negando as aparências\n...'

open("evidencias.txt").readlines()

['E nessa loucura de dizer que não te quero\n',
 'Vou negando as aparências\n',
 'Disfarçando as evidências\n',
 'Mas pra que viver fingindo\n',
 'Se eu não posso enganar meu coração\n',
 ...]
```

Abrindo e Processando Arquivos CSV

Vamos retomar o arquivo CSV do início deste roteiro, sem a linha de cabeçalho:

```
Hidrogênio, 1
Hélio, 2
Lítio, 3
Berílio, 4
Boro, 5
Carbono, 6
```

Suponha que este arquivo estivesse salvo no seu computador com nome `elementos.csv`. Nós poderíamos abrir este arquivo como fizemos na seção anterior, processar cada `string` individualmente e usar a função `split(",")` para separar cada linha em seus diversos campos. Porém, Python nos oferece a biblioteca `csv` que cuida desse processo e oferece outras facilidades para processamento de CSVs que não abordaremos nesta tarefa. Observe o código abaixo e seu resultado:

```
import csv
dados = csv.reader(open("elementos.csv"))

for linha in dados:
    print(linha)
```

```
['Hidrogênio', '1']
['Hélio', '2']
['Lítio', '3']
['Berílio', '4']
['Boro', '5']
['Carbono', '6']
```

A primeira linha é um comando `import` necessário para indicarmos ao Python que iremos utilizar a biblioteca `csv`. A variável `dados` agora contém um _iterador_ que nos permite "passear" por todas as linhas do CSV! Observe que trabalharemos dentro do loop com _listas de strings_ ao invés de `strings` simples, que precisariam ser divididas nos elementos. Isso é uma facilidade que usar a biblioteca CSV nos dá.

Descrição da entrada

Em cada teste, você lerá da entrada padrão dois números, um em cada linha. Esses números são os parâmetros `A` e `B`, respectivamente. Veja o exemplo abaixo:

```
15
-3600
```

Após receber os números, você deverá abrir o arquivo CSV, cujo nome será `dados.csv`. Este arquivo deverá ser colocado no mesmo diretório que o seu programa. No SuSy, uma cópia deste arquivo também estará no mesmo diretório que seu programa será testado. Cada linha conterá quatro `floats`, como comentado anteriormente.

```
<temperatura>,<pressão_medida>,<pressão_esperada>,ln(<pressão_esperada>)
```

Note que não há um dado extra com número de linhas nem uma linha com caracteres marcando o fim da entrada. Você deverá processar o arquivo de entrada até o final. Ao usar o iterador como exemplificado acima, isso já fica garantido.

Exemplo de linha do arquivo `dados.csv`:

```
"326.75","39.9","39.92","3.69"
```

Descrição da saída

A saída padrão irá acompanhar o tema desta tarefa e será em formato csv. Para cada linha do arquivo de entrada (`dados.csv`), você deve gerar uma linha na saída no formato abaixo. Note que todos os valores devem ser strings entre aspas duplas contendo números com duas casas decimais. Não deve haver espaços em branco entre os elementos ou ao final da linha.

```
<temperatura>,<pressão_medida>,<pressão_esperada>,<delta_pressao>,ln(<pressão_esperada>),<A_B/temperatura>,<delta_ln_for
```

Veja, a seguir, a descrição de cada um dos itens:

- `<temperatura>`: reprodução do valor da entrada.
- `<pressão medida>`: reprodução do valor da entrada.
- `<pressão esperada>`: reprodução do valor da entrada.
- `<delta_pressao>`: será obtido pela diferença `<pressão_medida> - <pressão_esperada>`.
- `ln(<pressão esperada>)`: reprodução do valor da entrada.
- `<A_B/temperatura>`: cálculo da fórmula a partir das constantes da entrada e do valor da `<A_B/temperatura>`.
- `<delta_ln_formula>`: será obtido pela diferença `<A_B/temperatura> - ln(<pressão esperada>)`

Exemplo de linha de saída:

```
"326.75","39.90","39.92","-0.02","3.69","3.98","0.29"
```

Testes com o SuSy

O conjunto de testes será formado por quatro testes abertos e um teste fechado. Todos os testes usarão o mesmo arquivo `dados.csv`. Os arquivos de teste irão variar o valor das constantes `A` e `B`.

Releia, se necessário, as instruções para fazer os testes em Testes com o SuSy.

Informações adicionais

Processamento de arquivos CSV é uma atividade bem comum em análise de dados uma vez que a maioria das ferramentas de coleta e processamento de dados é capaz de ler e escrever nesse formato. Ele também é bastante conveniente de ser processado, como vimos nessa tarefa. Algumas referências adicionais que se aprofundam nesse tópico são:

- Real Python - Reading and Writing CSV Files in Python
- Automate the Boring Stuff with Python

Agradecimentos

O enunciado desta tarefa foi elaborado com a colaboração de Felipe "Bidu", ex-PAD dessa disciplina e membro ativo da comunidade Python. Mais do trabalho dele pode ser encontrado em seu Github, https://github.com/fbidu. A doutoranda da FEQ, Aline Gallo De Mitri, colaborou com exemplos, dados e explicações sobre a parte referente ao experimento realizado.

Orientações para submissão

Veja aqui a página de submissão da tarefa. O arquivo a ser submetido deve se chamar `lab11.py`. No link Arquivos auxiliares há um arquivo `aux11.zip` que contém todos os arquivos de testes abertos e seus respectivos resultados compactados e o arquivo `dados.csv`.

O limite máximo será de 20 submissões. Serão considerados os resultados da última submissão.

O peso desta tarefa é 3.

O prazo final para submissão é 30/11/2019.

A nota desta tarefa é proporcional ao número de testes que executaram corretamente, desde que o código esteja coerente com o enunciado. **A submissão de um código que não implementa o algoritmo requisitado, mas que exibe as saídas esperadas dos testes abertos a partir da comparação de trechos da entrada será considerada fraude e acarretará a atribuição de nota zero à média final da disciplina.**

A imagem que ilustra esta tarefa foi obtida em https://towardsdatascience.com/scientific-data-analysis-pipelines-and-reproducibility.