Tarefa de Laboratório 02 - Repetição - Parte 1

Estatísticas do YouTube

A pandemia da COVID-19 alterou a rotina e o modo de comportamento de grande parte da população mundial. O aumento da frequência de acesso às plataformas de vídeos foi um exemplo disso.

Motivada por esses recentes aumentos, Beta quer investigar como o número de visualizações do conteúdo de um canal do YouTube se comportou nesse período. Para isso, ela anotou diariamente a quantidade de visualizações recebidas naquele canal nos últimos três anos e pediu sua ajuda para processar todas essas informações.

A sua tarefa neste lab será escrever um programa para computar algumas estatísticas (quantidade total, média e porcentagem) de visualizações de um canal no YouTube *nos últimos três anos* (2018, 2019 e 2020) a partir de um conjunto de anotações realizadas por Beta, em que cada uma contém a ***data da anotação*** (ano-mês-dia) e o ***número de visualizações*** naquela data.

Descrição da entrada

Nesta tarefa utilizaremos conceitos de estruturas de repetição e de acesso a índices de listas. A entrada que seu programa receberá é organizada da seguinte forma:

- A primeira linha da entrada possui o nome do canal.
- A segunda linha possui um número **N**, indicando a quantidade de anotações que serão lidas. Esse valor será sempre um inteiro maior do que ou igual a 3.
- Seguem **N** anotações, cada uma composta por 2 linhas:
    - A primeira linha de uma anotação possui uma data no formato `aaaa-mm-dd`, onde `aaaa` é o ano da anotação, `mm` é o mês e `dd` corresponde ao dia. Uma mesma data nunca se repetirá na entrada.
    - A segunda linha possui a quantidade de visualizações daquele dia. Esse valor sempre será um inteiro maior do que ou igual a 0.

Observe um exemplo de entrada abaixo:

```
pythonbrasil
5
2018-04-02
110
2020-11-09
110
2019-05-01
120
2018-09-17
70
2020-07-22
190
```

Considerações importantes para a entrada:

1. Infelizmente Beta é bastante desorganizada e registrou as anotações em qualquer ordem. Sendo assim, você não pode presumir que as anotações estarão ordenadas pela data.
2. Beta vem registrando anotações de visualização há muito tempo e não pôde separá-las para você, então você pode acabar encontrando anotações de anos anteriores à 2018 ou posteriores à 2020.
3. Sempre haverá pelo menos uma anotação para cada um dos três últimos anos (2018, 2019 e 2020).

Descrição da saída

A saída do seu programa deve conter 18 linhas, sem acentos, organizadas da seguinte forma:

- **Canal**: nome do canal.
- **Total de views do trienio**: total de visualizações recebidas ao longo dos **três últimos anos**.
- **Media de views do trienio**: média de visualizações ao longo dos **três últimos anos** formatada com duas casas decimais. Essa média deve ser calculada em relação à quantidade de dias que apareceram na entrada **para o triênio**, isto é, *media_trienio=views_trienio/anotacoes_trienio.*
- *(Linha em branco)*
- **2018**
- **Total**: total de visualizações recebidas em **2018**.
- **Porcentagem das views do trienio**: porcentagem de visualizações recebidas em **2018**, em relação ao total do triênio, formatada com 2 casas decimais.
- **Media:** média das visualizações recebidas em **2018** formatada com 2 casas decimais.
- *(Linha em branco)*
- **2019**
- **Total**: total de visualizações recebidas em **2019**.
- **Porcentagem das views do trienio**: porcentagem de visualizações recebidas em **2019**, em relação ao total do triênio, formatada com 2 casas decimais.
- **Media:** média das visualizações recebidas em **2019** formatada com 2 casas decimais.
- *(Linha em branco)*
- **2020**
- **Total**: total de visualizações recebidas em **2020**.
- **Porcentagem das views do trienio**: porcentagem de visualizações recebidas em **2020**, em relação ao total do triênio, formatada com 2 casas decimais.
- **Media:** média das visualizações recebidas em **2020** formatada com 2 casas decimais.

A saída correspondente ao exemplo de entrada fornecido é:

```
Canal: pythonbrasil
Total de views do trienio: 600
Media de views do trienio: 120.00

2018
Total: 180
Porcentagem das views do trienio: 30.00
Media: 90.00

2019
Total: 120
Porcentagem das views do trienio: 20.00
Media: 120.00

2020
Total: 300
Porcentagem das views do trienio: 50.00
Media: 150.00
```

Obs.: para calcular o valor da média, você deve levar em conta somente a quantidade de dias que apareceram na entrada. Se houve, por exemplo, somente três anotacões no ano de 2018 com números de visualizações `10`, `20` e `30`, a média de visualizações de 2018 será dada por `(10+20+30)/3 = 20.00`.

E se o total de visualizações no triênio for zero?

Nesse caso, o percentual do total de views a que cada um dos três últimos anos corresponde será dado por `0/0`, o que chamamos de uma *indeterminação*. Por conta disso, para essa tarefa, sempre que o total de views do triênio for zero, você deverá imprimir **indeterminada** como resultado da porcentagem de views de cada ano. Consulte o caso de teste 4 no fim desse enunciado para ver um exemplo.

Cuidado: se você tentar realizar a divisão de algum número por 0 no seu programa, ele retornará um erro. Veja um exemplo abaixo:

```
>>> x = 0/0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
```

Dicas para esta tarefa

Preparamos algumas dicas para ajudar você nesse lab, veja abaixo.

Como obter o ano a partir da data?

Após ler a linha correspondente à data, podemos separar o ano através da função `split()`, indicando uma string separadora (no nosso caso o hífen). Por exemplo:

```
data = input().split('-')
```

O comando `split()` retorna uma lista com os trechos da string que foram separados pela sequência de caracteres especificada como parâmetro. Para operar com o ano, podemos acessar o primeiro elemento da lista escrevendo `data[0]`. Observe o exemplo abaixo:

```
data_formatada = '2021-04-03'
data_separada = data_formatada.split('-')
print('Ano:', data_separada[0])
print('Mes:', data_separada[1])
print('Dia:', data_separada[2])
```

Ao executar o exemplo, a seguinte resposta é obtida:

```
Ano: 2021
Mes: 04
Dia: 03
```

Importante: não será permitida a utilização de bibliotecas externas, nem para lidar com a data (como a datetime), tampouco com o cálculo das estatísticas (como NumPy, SciPy, statistics, etc.).

Como utilizar os casos de teste abertos para testar o programa?

Quando executamos nosso programa pelo terminal, todo comando de entrada do usuário (*user input*), como o `input()` do `python`, lê os dados a partir do teclado. Nesse caso, costumamos dizer que o teclado é o *dispositivo de entrada padrão*. Podemos alterar esse dispositivo padrão utilizando o operador `<` em alguns terminais. Isso significa dizer que o comando `input()` do nosso programa poderá, por exemplo, buscar automaticamente os dados a serem lidos a partir de um arquivo de texto, como os casos de teste desse lab.

De forma análoga, quando utilizamos um comando de escrita, como o `print()`, os dados são impressos na tela do computador através do terminal, que é o *dispositivo de saída padrão*. Também podemos alterar esse dispositivo em alguns terminais através do operador `>`. Assim será possível escrever a saída do nosso programa em um arquivo de texto.

Vamos ilustrar melhor como isso funciona abaixo:

1. Depois que seu programa estiver pronto para ser testado, abra um terminal (como o `bash` no *Linux* ou no *macOS*, ou o `cmd` no *Windows*) e navegue até a pasta onde estão o código do seu programa e os casos de teste (você pode usar o comando `cd` para isso).
2. Execute seu programa lendo os dados do arquivo de entrada através do comando `python3 codigo.py < x.in`, onde `codigo.py` é o arquivo de código do seu programa e `x.in` é a entrada de um caso de teste. Observe um exemplo para o primeiro caso de teste dessa tarefa:

    ```
    >>> python3 codigo.py < 1.in
    Canal: Unicamp IC
    Total de views do trienio: 200
    Media de views do trienio: 66.67

    2018
    Total: 100
    Porcentagem das views do trienio: 50.00
    Media: 100.00

    2019
    Total: 50
    Porcentagem das views do trienio: 25.00
    Media: 50.00

    2020
    Total: 50
    Porcentagem das views do trienio: 25.00
    Media: 50.00
    ```

    Obs.: os caracteres ">>>" são do próprio terminal.

3. Você também pode salvar a saída do seu programa em arquivo de texto através do comando `python3 codigo.py < x.in > x.myout`, onde `x.myout` é o nome do arquivo que será gerado com a saída do programa. Observe um exemplo:

    ```
    >>> python3 codigo.py < 1.in > 1.myout
    ```

Após ter gerado um arquivo com a saída do seu programa, você pode utilizar um comando do próprio terminal para verificar se ela é idêntica à saída esperada para o caso de teste.

Para isso, no ***Linux*** ou no ***macOS*** você pode utilizar o comando **diff** escrevendo `diff --strip-trailing-cr x.out x.myout`, onde `x.out` é a saída esperada do caso de teste e `x.myout` é a saída do seu programa. Caso não haja diferença entre os arquivos, nada será impresso. Note que foi especificada a opção `--strip-trailing-cr` no comando, ela é responsável por ignorar diferenças entre o formato de quebra de linha do *Windows* e do *Linux*. Essa opção é necessária, pois o run.codes formata a quebra de linha conforme o padrão do *Windows*. Exemplo:

```
>>> diff --strip-trailing-cr 1.out 1.myout
>>>
```

Já no `cmd.exe` do ***Windows*** você pode realizar a comparação usando o comando **fc** escrevendo `fc x.out x.myout`, onde `x.out` é a saída esperada do caso de teste e `x.myout` é a saída do seu programa. Caso os arquivos não possuam diferenças, a mensagem *"FC: nenhuma diferença encontrada"* será impressa. Exemplo:

```
C:\Casos de Teste> fc 1.out 1.myout
Comparando os arquivos 1.out e 1.MYOUT
FC: nenhuma diferença encontrada
```

Observações:

1. O `powershell` do *Windows* não reconhece os operadores `<` e `>` como comandos de redirecionamento, portanto, opte por utilizar o terminal `cmd.exe`.
2. A saída dos comandos de comparação `fc` e `diff` podem ser um pouco difíceis de se interpretar no início, portanto, recomendamos que você utilize os casos de teste somente quando acreditar que seu programa já está correto.
3. Você também pode consultar esse vídeo do prof. Sidney Nogueira da UFRPE como um exemplo.