Enunciados dos Trabalhos para Casa (TPC)

da UC de **Processamento de Linguagens**, ano Letivo 2022/23 , 3º ano de LEI

Relativos à 1º parte do programa: Filtros de Texto baseados em ER **texto em negrito**

TPC1: Análise de dados: doença cardíaca

(publicado em **2023.02.14**)

Descarregue o ficheiro de dados: `myheart.csv` Crie um programa em Python, conjunto de funções, que responda às seguintes questões:

- Crie uma função que lê a informação do ficheiro para um modelo, previamente pensado em memória;
- Pense num modelo para guardar uma distribuição;
- Crie uma função que calcula a distribuição da doença por sexo;
- Crie uma função que calcula a distribuição da doença por escalões etários. Considere os seguintes escalões: [30-34], [35-39], [40-44], ...
- Crie uma função que calcula a distribuição da doença por níveis de colesterol. Considere um nível igual a um intervalo de 10 unidades, comece no limite inferior e crie os níveis necessários até abranger o limite superior;
- Crie uma função que imprime na forma de uma tabela uma distribuição;
- Especifique um programa que ao executar apresenta as tabelas correspondentes às distribuições pedidas;
- Extra: explore o módulo matplotlib e crie gráficos para as suas distribuições.

TPC2: Somador on/off

(publicado em **2023.02.21**)

Crie um programa em Python que tenha o seguinte comportamento:

- Pretende-se um programa que some todas as sequências de dígitos que encontre num texto;
- Prepare o programa para ler o texto do canal de entrada: stdin;
- Sempre que encontrar a string “Off” em qualquer combinação de maiúsculas e minúsculas, esse comportamento é desligado;
- Sempre que encontrar a string “On” em qualquer combinação de maiúsculas e minúsculas, esse comportamento é novamente ligado;
- Sempre que encontrar o caráter “=”, o resultado da soma é colocado na saída.

TPC3: Processador de Pessoas listadas nos Róis de Confessados

(publicado em **2023.02.28**)

Construa agora um ou vários programas Python para processar o texto 'processos.txt' (procurar o ficheiro no Bb) com o intuito de calcular frequências de alguns elementos (a ideia é utilizar arrays associativos, dicionários em Python, para o efeito) conforme solicitado a seguir:

> a) Calcula a frequência de processos por ano (primeiro elemento da data);

> b) Calcula a frequência de nomes próprios (o primeiro em cada nome) e apelidos (o ultimo em cada nome) por séculos e apresenta os 5 mais usados;

> c) Calcula a frequência dos vários tipos de relação: irmão, sobrinho, etc.;

> d) Converta os 20 primeiros registos num novo ficheiro de output mas em formato **Json**.

TPC4: Ficheiros CSV com listas e funções de agregação

(publicado em **2023.03.06**)

Cria um programa em Python que implementa um conversor de um ficheiro CSV (Comma separated values) para o formato JSON. Para se poder realizar a conversão pretendida, é importante saber que a primeira linha do CSV dado funciona como cabeçalho que define o que representa cada coluna. Por exemplo, o seguinte ficheiro "`alunos.csv`":

```
Número,Nome,Curso
3162,Cândido Faísca,Teatro
7777,Cristiano Ronaldo,Desporto
264,Marcelo Sousa,Ciência Política
```

Que corresponde a uma tabela com 3 registos de informação: a primeira linha de cabeçalho identifica os campos de cada registo, `Número`, `Nome`, `Curso`, e as linhas seguintes contêm os registos de informação.

No entanto, os CSV recebidos poderão conter algumas extensões cuja semântica se explica a seguir:

1. Listas

No cabeçalho, cada campo poderá ter um número `N` que representará o número de colunas que esse campo abrange. Por exemplo, imaginemos que ao exemplo anterior se acrescentou um

campo `Notas`, com `N = 5` ("`alunos2.csv`"):

```
Número,Nome,Curso,Notas{5},,,,,
3162,Cândido Faísca,Teatro,12,13,14,15,16
7777,Cristiano Ronaldo,Desporto,17,12,20,11,12
264,Marcelo Sousa,Ciência Política,18,19,19,20,18
```

Isto significa que o campo Notas abrange 5 colunas (reparem que se colocaram os campos que sobram a vazio, para o **CSV bater certo**).

2. Listas com um intervalo de tamanhos

Para além de um tamanho único, podemos também definir um intervalo de tamanhos `{N, M}`, significando que o número de colunas de um certo campo pode ir de `N` até `M` ("`alunos3.csv`"):

```
Número,Nome,Curso,Notas{3,5},,,,,
3162,Cândido Faísca,Teatro,12,13,14,,
7777,Cristiano Ronaldo,Desporto,17,12,20,11,12
264,Marcelo Sousa,Ciência Política,18,19,19,20,
```

À semelhança do ponto anterior, havendo colunas vazias, os separadores têm de estar lá, o número de colunas deverá ser sempre igual ao valor máximo de colunas, poderão é estar preenchidas com informação ou não.

3. Funções de agregação

Para além de listas, podemos ter funções de agregação, aplicadas a essas listas. Veja os seguintes exemplos: "`alunos4.csv`"

```
Número,Nome,Curso,Notas{3,5}::sum,,,,,
3162,Cândido Faísca,Teatro,12,13,14,,
7777,Cristiano Ronaldo,Desporto,17,12,20,11,12
264,Marcelo Sousa,Ciência Política,18,19,19,20,
```

e "`alunos5.csv`":

```
Número,Nome,Curso,Notas{3,5}::media,,,,,
3162,Cândido Faísca,Teatro,12,13,14,,
7777,Cristiano Ronaldo,Desporto,17,12,20,11,12
264,Marcelo Sousa,Ciência Política,18,19,19,20,
```

Resultados esperados

O resultado final esperado é um ficheiro JSON resultante da conversão dum ficheiro CSV. Por exemplo, o ficheiro "`alunos.csv`" (original), deveria ser transformado no seguinte ficheiro "`alunos.json`":

```
[
  {
    "Número": "3612",
    "Nome": "Cândido Faísca",
    "Curso": "Teatro"
  },
  {
    "Número": "7777",
    "Nome": "Cristiano Ronaldo",
    "Curso": "Desporto"
  },
  {
    "Número": "264",
    "Nome": "Marcelo Sousa",
    "Curso": "Ciência Política"
  }
]
```

No caso de existirem listas, os campos que representam essas listas devem ser mapeados para listas em JSON ("`alunos2.csv ==> alunos2.json`"):

```
[
  {
    "Número": "3612",
    "Nome": "Cândido Faísca",
    "Curso": "Teatro",
    "Notas": [12,13,14,15,16]
  },
  {
    "Número": "7777",
    "Nome": "Cristiano Ronaldo",
    "Curso": "Desporto",
    "Notas": [17,12,20,11,12]
  },
  {
    "Número": "264",
    "Nome": "Marcelo Sousa",
    "Curso": "Ciência Política",
    "Notas": [18,19,19,20,18]
  }
]
```

Nos casos em que temos uma lista com uma função de agregação, o processador deve executar a função associada à lista, e colocar o resultado no JSON, identificando na chave qual foi a função executada ("`alunos4.csv ==> alunos4.json`"):

```
[
  {
    "Número": "3612",
    "Nome": "Cândido Faísca",
    "Curso": "Teatro",
    "Notas_sum": 39
  },
  {
    "Número": "7777",
    "Nome": "Cristiano Ronaldo",
    "Curso": "Desporto",
    "Notas_sum": 72
  },
  {
    "Número": "264",
    "Nome": "Marcelo Sousa",
    "Curso": "Ciência Política",
    "Notas_sum": 76
  }
]
```

Outros...

Se tiverem tempo e vontade podem sempre estender este enunciado acrescentando outras funções de agregação (maior, menor, ...), a possibilidade de termos mais de uma função de agregação em simultâneo num ficheiro, ... a creatuvidade no seu melhor...