Universidade do Minho

Processsamento de Linguagens e Compiladores (3º ano de Curso)

Trabalho Prático 1 - Grupo 3

Relatório de Desenvolvimento

André Neves da Costa
(A95869)

Tiago Emanuel Lemos Teixeira
(A97666)

13 de novembro de 2022

Resumo

Isto é o relatório do trabalho prático 1 de Processsamento de Linguagens e Compiladores, onde vai ser explicado o método de trabalho por trás do seu desenvolvimento.

Conteúdo

1 Introdução **3**

2 Enunciados dos problemas **4**
2.1 Processador de Registos de Exames Médicos Desportivos 4
2.2 Ficheiros CSV com listas e funções de agregação 5
2.2.1 Listas 5
2.2.2 Funções de Agregação 5

3 Decisões Tomadas **7**
3.1 EX 2 - Processador de Registos de Exames Médicos Desportivos 7
3.1.1 Página Principal 8
3.1.2 Indicadores Estatísticos 9
3.1.3 Representação das Distribuições 11
3.2 EX 5 - Ficheiros CSV com listas e funções de agregação 18
3.2.1 Conversão de CSV para JSON 18
3.2.2 Listas com tamanho definido 19
3.2.3 Listas com um intervalo de tamanhos 20
3.2.4 Funções de agregação 22
3.2.5 Extra - Escolha do ficheiro e pesquisa 24

4 Exemplos de utilização **28**
4.1 EX 2 - Processador de Registos de Exames Médicos Desportivos 28
4.2 EX 5 - Ficheiros CSV com listas e funções de agregação 34
4.2.1 Exemplo 1 34
4.2.2 Exemplo 2 34
4.2.3 Exemplo 3 34
4.2.4 Exemplo 4 34
4.2.5 Exemplo Extra 34
4.2.6 Outputs 35

5 Conclusão **39**

A Código dos Programas **40**
A.1 Processador de Registos de Exames Médicos Desportivos 40

A.2 Ficheiros CSV com listas e funções de agregação 46

Capítulo 1

Introdução

No âmbito da disciplina de Processamento de Linguagens e Compiladores foi-nos proposto pelo docente Pedro Manuel Rangel Santos Henriques um trabalho cuja realização tem os seguintes objetivos:

- aumentar a capacidade de escrever Expressões Regulares (ER) para descrição de padrões de frases dentro de textos;
- desenvolver, a partir de ER, sistematicamente Processadores de Linguagens Regulares, ou Filtros de Texto (FT), que filtrem ou transformem textos com base no conceito de regras de produção Condição-Ação;
- utilizar o módulo 're' com suas funções de search(), split(), sub() do Python para implementar os FT pedidos.

Neste trabalho, foi nos dada a oportunidade de, em 5 problemas, escolher pelo menos um deles para desenvolver.

As 5 questões colocadas a todos os grupos de trabalho são as seguintes:

1. Processador de Pessoas listadas nos Róis de Confessados;
2. **Processador de Registos de Exames Médicos Desportivos;**
3. EnameXPro, processador de Enamex;
4. BibTeXPro, Um processador de BibTeX;
5. **Ficheiros CSV com listas e funções de agregação.**

Com a disponibilidade que tivemos, tomamos liberdade de resolver, não um, mas dois dos problemas propostos. Sendo esses problemas, os que se encontram destacados a negrito. Salientamos que também resolvemos desafios que nós próprios propusemos, para adicionar algo além do que cada tópico pedia.

Capítulo 2

Enunciados dos problemas

2.1 Processador de Registos de Exames Médicos Desportivos

Neste exercício pretende-se trabalhar com um dataset gerado no âmbito do registo de exames médicos desportivos. Construa, então, um ou vários programas Python para processar o dataset "emd.csv"e produzir o solicitado nas alíneas seguintes:

- Criar um website com as seguintes caraterísticas:
  1. Página principal: de nome "index.html", contendo os seguintes indicadores estatísticos:
     a) Datas extremas dos registos no dataset;
     b) Distribuição por modalidade em cada ano e no total;
     c) Distribuição por idade e género (para a idade, considera apenas 2 escalões: < 35 anos e ≥ 35);
     d) Distribuição por morada;
     e) Percentagem de aptos e não aptos por ano.
  2. Página do indicador: clicando no indicador na página principal, devemos saltar para a página do indicador onde temos a informação que permitiu obter esse indicador. Por exemplo, para a distribuição por morada, a página deverá apresentar uma lista de moradas, ordenada alfabeticamente e para cada morada deverá apresentar uma sublista de registos, ordenada alfabeticamente por nome de atleta (com os dados: nome do atleta, modalidade).

2.2 Ficheiros CSV com listas e funções de agregação

Neste enunciado pretende-se fazer 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
```

No entanto, neste trabalho, os **CSV** recebidos têm algumas extensões.

2.2.1 Listas

Nestes datasets, poderemos ter conjuntos de campos que formam listas.

Listas com tamanho definido

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
```

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,
```

2.2.2 Funções de Agregação

Para além de listas, podemos ter funções de agregação, aplicadas a essas listas. Veja os seguintes exemplos ("alunos4.csv"e "alunos5.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,
```

```
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,
```

Capítulo 3

Decisões Tomadas

3.1 EX 2 - Processador de Registos de Exames Médicos Desportivos

A partir dos dados que nos são apresentados no dataset **"emd.csv"**, neste exercício, teremos que distribuir tais dados, usando a linguagem ***html***, para obtermos o resultado esperado.

No decorrer deste problema, tivemos que dividí-lo em 3 partes, para que a resolução do mesmo seja facilitada:

1. Criação da página principal (index.html);
2. Criação de indicadores estatísticos, que levarão a outros ficheiros ***html*** com as respetivas distribuições;
3. Desenvolvimento das páginas dos indicadores, onde vão ficar representadas as distribuições em questão;
   (a) Distribuição por Modalidade;
   (b) Distribuição por Idade e Género;
   (c) Distribuição por Morada;
   (d) Percentagens de Aptabilidade.
4. Melhorar a apresentação das distribuições, colocando-as de forma tabelada.

3.1.1 Página Principal

Este código é o começo do exercício 2. Servirá para criar a página inicial, onde serão expostos os indicadores para as diferentes distribuições que o exercício requer que resolvamos. Também é aqui que vamos abrir o ficheiro **"emd.csv"** que contém os dados sobre os atletas, que serão posteriormente analisados e distribuídos. Ficheiro esse que vai ser transformado numa lista de strings (implementando a função "*readlines*"), e guardado numa variável que irá ser usada nas etapas seguintes.

```
1 import re, os
2
3 # Abrir ficheiros ----------------------
4 try:
5   fhtml = open("index.html","x")
6 except FileExistsError:          # P gina Principal
7   os.remove("emd.html")
8   fhtml = open("index.html","x")
9
10 f = open("C:\\Users\\letsd\\OneDrive\\ rea de Trabalho\\uni\\PLCTP\\emd.csv", 'r')
11
12
13 # Separar as linhas de emd em lista de strings
14
15 csvreader = f.readlines()
16
17 # message vai guardar todo o c digo html de index.html
18
19 message = "<html><head><style>a{font-size: 18px; color: black;}h1{font-size: 32px;}h2{
    font-size: 24px;} li{margin-bottom: 15px;}</style></head><body><h1>Registos de
    Exames M&eacute;dicos Desportivos</h1><ul>"
```

3.1.2 Indicadores Estatísticos

A variável "**anos**"comtém a lista de anos, desde a data do primeiro atleta existente no dataset até à data do último. Vai ser importante no código deste exercício, porque vai envolver-se nas distribuições que serão abordadas mais para a frente.

Como podemos ver, aqui é usada a função *re.search*, para pesquisar o ano em que o atleta se inscreveu. Esta função requer a implementação de expressões regulares, e sendo assim, achamos a expressão $[0-9]${4}(?=(-$[0-9]${2}-$[0-9]${2})) mais apropriada para o que queriamos encontrar.

Expressão regular utilizada

- $[0-9]${4}(?=(-$[0-9]${2}-$[0-9]${2}))
  - $[0-9]${4} → Faz correspondência a uma sequência de 4 digitos;
  - **(?=(-**$[0-9]${2}-$[0-9]${2}**))** → Faz correspondência a, se tiver na posição seguinte ao que procuramos, uma sequência de 2 digitos entre duas ocorrências de '-', seguido de mais uma sequência de 2 digitos;

```
1 # Datas extremas dos registos no dataset ----------------------------------
2
3 anos = set()
4 for line in csvreader:
5   n = re.search(r"\d{4}(?=(-\d{2}-\d{2}))",line)
6   if n:
7     anos.add(n.group(0))
8 anos = sorted(list(anos))
9
10 message += "<h2>Dados registados entre " + anos[0] + " e " + anos[-1] + "</h2>"
```

Nos seguintes quatro excertos de código, é mostrado como se criam as ligações para outros ficheiros ***html***. Cada excerto terá uma variável que terá uma string que vai aumentar gradualmente de acordo a quantidade de dados que o ficheiro "**emd.csv**"tem. No fim de cada um destes códigos, será adicionada na variável "**message**"os indicadores a que lhes corresponde.

(linha 33)

```
1 #  Distribui  o por modalidade em cada ano e no total
     ---------------------------------------------------
2
3 try:
4   fdmod = open("DistribModalidade.html","x")            #abrir ficheiro html
5 except FileExistsError:
6   os.remove("DistribModalidade.html")
7   fdmod = open("DistribModalidade.html","x")
8
9 messagemodalidade = "<html><head></head><body><h1>Distribui&ccedil;&atilde;o por
      modalidade</h1><ol>"
10
11 message+= "<li><a href = \"DistribModalidade.html\">Distribui&ccedil;&atilde;o por
      modalidade em cada ano</a></li>"
```

```
12 #criar liga   o do indicador
```

(linha 117)

```
1 #  Distribui   o por idade e g nero
     ----------------------------------------------------------------------

2
3 try:
4    fdig = open("DistribIG.html","x")
5 except FileExistsError:
6    os.remove("DistribIG.html")
7    fdig = open("DistribIG.html","x")
8
9 messageig = "<html><head></head><body><h1>Distribui&ccedil;&atilde;o por idade e g&
      eacute;nero</h1><ol>"
10 message+= "<li><a href=\"DistribIG.html\">Distribui&ccedil;&atilde;o por idade e g&
      eacute;nero</a></li>"
```

(linha 193)

```
1 # Distribui   o por morada
     ----------------------------------------------------------------------

2
3 message+="<li><a href=\"DistribMorada.html\">Distribui&ccedil;&atilde;o por morada</a
      ></li>"
4 try:
5    fdm = open("DistribMorada.html","x")
6 except FileExistsError:
7    os.remove("DistribMorada.html")
8    fdm = open("DistribMorada.html","x")
9
10 messagemorada = "<html><head><style>h1{font-size: 32px;}</style></head><body><h1>
      Distribui&ccedil;&atilde;o por morada</h1><ol>"
```

(linha 242)

```
1 # Percentagem de aptos e n o aptos por ano
     ----------------------------------------------------------------------

2
3 message+="<li><a href=\"Aptidao.html\">Percentagem de aptos e n&atilde;o aptos</a></
      li>"
4 try:
5    fapt = open("Aptidao.html","x")
6 except FileExistsError:
7    os.remove("Aptidao.html")
8    fapt = open("Aptidao.html","x")
9
10 messageapt = "<html><head><style>h1{font-size: 32px;}</style></head><body><h1>
      Percentagem de aptos e n&atilde;o aptos</h1><ol>"
```

3.1.3 Representação das Distribuições

Distribuição de modalidades por ano e no total

Aqui organizaremos os dados do dataset numa tabela em função das modalidades existentes e dos anos. A variável responsável por escrever no ficheiro "**DistribModalidade.html**", será a "**messagemodalidade**". Nela conterá os valores referentes de forma tabelada.

O processo para procurar por todas as modalidades, usaremos a função "**re.search**"e a expressão "**(?<=,**$[MF]$**,)(**$[A-Z][a-z]+$**),(**$[A-Z][\hat{},]+$**)**", colocando-as na variável modalidades (lista de strings), ordenada alfabeticamente.

Expressão regular utilizada

- **(?<=,**$[MF]$**,)(**$[A-Z][a-z]+$**),(**$[A-Z][\hat{},]+$**))**
  - **(?<=,**$[MF]$**,)** → Faz correspondência a, se existir um M ou F entre duas ',', na posição anterior;
  - $([A-Z][a-z]+)$**,** → Faz correspondência a uma letra maiúscula e uma ou mais minúsculas, seguida de uma ',';
  - $([A-Z][\hat{},]+)$ → Faz correspondência a uma letra maiúscula seguida de um ou mais caracteres, excetuando a ','
- $[0-9]$**{4}(?=(-**$[0-9]$**{2}-**$[0-9]$**{2}))** (já usada nas datas extremas);

```
1
2 modalidades = set()
3 # for ir   procurar todas as modalidades existentes no dataset
4 for line in csvreader[1:]:
5   n = re.search(r"(?<=,[MF],)([A-Z][a-z]+),([A-Z][^,]+)",line)
6   if n:
7     modalidades.add(n.group(2))
8
9 modalidades = sorted(list(modalidades))
10
11 messagemodalidade+="<div class=\"container\">"
12
13 messagemodalidade += "<div class=\"sub-container\">"
14 for ano in anos:
15   messagemodalidade+= "<p><strong>" + ano + ":</strong>" + "</p>"
16 messagemodalidade+= "<p><strong>Total:</strong></p></div>"
```

A seguir, faremos algo que não foi pedido no exercício, mas tomamos iniciativa para tal. Nesta parte do código, será criada uma tabela que mostrará o número de atletas presentes nas modalidades em cada ano e no total (linhas: anos; colunas = modalidades). Percorreremos a lista de modalidades e escreveremos por linha o número de atletas usando a variável **counter**. Em cada iteração, lemos os dados do **csvreader**, procurando por:

- anos ($[0-9]$**{4}(?=(-**$[0-9]$**{2}-**$[0-9]$**{2}))**) - para fazer comparação com o ano em questão;

- modalidade ((**?<=,**$[MF]$**,)(**$[A-Z][a-z]+$**),(**$[A-Z][^,]+$**)))**) - para fazer comparação com a modalidade em questão;

Se estes dados forem devidamente encontrados, então a variável **counter** será incrementada. Repetindo este processo até as modalidades acabarem, a tabela deve estar organizada como esperado.

```

for i in modalidades:
  messagemodalidade += "<div class=\"sub-container\"><head><strong>" + i + "</strong></head>"
  total = 0
  for ano in anos:
    counter = 0
    messagemodalidade+="<p>"
    for line in csvreader:
      nanos = re.search(r"\d{4}(?=(-\d{2}-\d{2}))",line)
      nmodalidade = re.search(r"(?<=,[MF],)([A-Z][a-z]+),([A-Z][^,]+)",line)
      if nanos and nmodalidade and nanos.group(0) == ano and nmodalidade.group(2) == i:
        counter+=1
    total+=counter
    messagemodalidade+= str(counter) + "</p>"
  messagemodalidade+= "<p>" + str(total) + "</p></div>"
messagemodalidade+="</div>"

messagemodalidade = messagemodalidade[:60] + "<style>h1{font-size: 32px;}.container {display: flex; gap: 10px; align-items: flex-end;}</style>" + messagemodalidade[60:]

messagemodalidade+="<p>----------</p>"
```

No próximo excerto de código, mostrará o que o exercício pretendia que os alunos fizessem. Percorremos **csvreader** para procurar as moradas ((**?<=,**$[MF]$**,)(**$[A-Z][a-z]+$**),(**$[A-Z][^,]+$**))**, anos e nomes ((**?<=**$[0-9]${**4**}-$[0-9]${**2**}-$[0-9]${**2**},)(**$[A-Z][a-z]+$**),(**$[A-Z][a-z]+$**),**), e caso encontrar colocar no dicionário **nmod** (keys: (modalidade,ano); valores: nome).

```

# Guarda-se os dados sobre modalidades num dicionrio, (modalidade, ano) : nome

nmod = {}

for line in csvreader:
  moradaemodal_dm = re.search(r"(?<=,[MF],)([A-Z][a-z]+),([A-Z][^,]+)",line)
  anos_dm = re.search(r"\d{4}(?=(-\d{2}-\d{2}))",line)
  nomes_dm = re.search(r"(?<=\d{4}-\d{2}-\d{2},)([A-Z][a-z]+),([A-Z][a-z]+),",line)

  if moradaemodal_dm and nomes_dm and anos_dm:
    if (moradaemodal_dm.group(2),anos_dm.group(0)) not in nmod:
      nmod[(moradaemodal_dm.group(2),anos_dm.group(0))] = [nomes_dm.group(1) + " " + nomes_dm.group(2)]
    else:
      nmod[(moradaemodal_dm.group(2),anos_dm.group(0))].append(nomes_dm.group(1) + " " + nomes_dm.group(2))
```

Por último, é escrito num estilo indexado, por ano, todas as modalidades e todos os seus participantes. (ver figura 4.2)

```
1
2
3 # modalidades j  definida
4 modalidadesord = sorted(modalidades)
5 # anos j  definida nas datas extremas
6
7 # a partir do dicion rio escreve-se o c digo html em que listamos os atleta por
      modalidade e ano
8
9 for ano in anos:
10   messagemodalidade += "<p><strong>" + ano + "</strong></p><ul>"
11   for mod in modalidadesord:
12     messagemodalidade += "<li><strong>" + mod + "</strong><ol>"
13     for atletas in list(nmod.items()):
14       if atletas[0][1] == ano and atletas[0][0] == mod:
15         atlord = sorted(atletas[1])
16         for atleta in atlord:
17           messagemodalidade += "<li>" + atleta + "</li>"
18     messagemodalidade+= "</ol></li>"
19   messagemodalidade+="</ul>"
20
21
22 fdmod.write(messagemodalidade)
```

Expressão regular utilizada

- **r"(?<=,)"+i+r"(?=,)**
  - **(?<=,)** → Faz correspondência a, se existir uma ',' na posição anterior;
  - **i** → Faz correspondência a $[MF]$ (um 'M' ou 'F');
  - **(?=,)** → Faz correspondência a, se existir uma ',' na posição seguinte.
- **(?<=,)**$[0-9]${**2**}**(?=,**$[MF]$**)**
  - $[0-9]${**2**} → Faz correspondência a uma sequência de 2 digitos;
  - **(?=,**$[MF]$**)** → Faz correspondência a, se existir uma ',' seguida de um 'M' ou 'F'.
- **(?<=,)**$[MF]$**(?=,)**
  - $[MF]$ → Faz correspondência a um 'M' ou 'F';
- **(?<=**$[0-9]${**4**}**-**$[0-9]${**2**}**-**$[0-9]${**2**}**,)(**$[A-Z][a-z]+$**),(**$[A-Z][a-z]+$**),**
  - **(?<=**$[0-9]${**4**}**-**$[0-9]${**2**}**-**$[0-9]${**2**}**,)** → Faz correspondência a, se existir uma data seguida de uma ',' na posição anterior ;
  - **(**$[A-Z][a-z]+$**),(**$[A-Z][a-z]+$**),** → Faz correspondência a duas ocorrências de: uma letra maiúscula seguida de uma ou mais letras minúsculas e uma ','.

Distribuição por idade e género

Como foi feito nas modalidades, fizemos uma tabela com a quantidade de atletas em função do seu género e a sua idade. As variáveis **countmenor** e **countmaior**, serão incrementadas consoante a pessoa tenha menos de 35 anos ou tenha mais que 35 anos, respetivamente. Estas variáveis vão ser comparadas com a idade do atleta que foi encontrada no **csvreader**, usando a expressão regular **(?<=,)**[0−9]**{2}(?=,**[MF]**)**. Da mesma forma o género da pessoa pesquisada no ficheiro (**(?<=,)"+i+r"(?=,)**) será comparado com a variável **i** que é 'M' ou 'F'.

A **messageig**, será responsável por escrever no ficheiro **DistribIG.html** tudo o que vai acumular no decorrer destes dois excertos.

```

messageig+= "<div class=\"container\"><div class=\"sub-container\"><p><strong>Menos de 35 anos:</strong></p>"
messageig+="<p><strong>Mais ou com 35 anos:</strong></p></div>"


# Constru   o da tabela com a distribui   o

for i in ['M','F']:
  countmenor=0
  countmaior=0
  messageig+= "<div class=\"sub-container\"><p>"
  for line in csvreader:
    genero = re.search(r"(?<=,)"+i+r"(?=,)",line)
    idade = re.search(r"(?<=,)\d{2}(?=,[MF])",line)
    if genero and idade and genero.group(0) == i and int(idade.group(0)) < 35:
      countmenor +=1
    elif genero and idade and genero.group(0) == i and int(idade.group(0)) >= 35:
      countmaior +=1
  messageig+= "<strong>" + i + "</strong>" + "</p><p>" + str(countmenor) + "</p><p>" + str(countmaior) + "</p></div>"

messageig+="</div>"

messageig = messageig[:60] + "<style>h1{font-size: 32px;}.container{display:flex;gap: 10px;align-items: flex-end;}</style>" + messageig[60:]

messageig+="<p>————————————————————————————————</p>"
```

No entanto, no que toca ao exercício em si. Neste excerto, será escrito o código, de modo a escrever no ficheiro do indicador as informações indexadas como demonstrado na figura 4.3.

As variáveis correspondentes às idades, géneros e nomes dos atletas, são obtidas a partir do **csvreader**. As mesmas vão ser usadas para verificar se essas informações estão presentes no dicionário **dictig** (keys: (género, idade); valores: nome).

No fim, o **dictig** vai ser percorrido, e os dados acumulados na variável **messageig**

```
# Guarda-se os dados sobre idade e g nero num dicion rio , (genero, idade) : nome

dictig = {}
```

```
4
5 for line in csvreader:
6   generos_ig = re.search(r"(?<=,)[MF](?=,)",line)
7   idades = re.search(r"(?<=,)\d{2}(?=,[MF])",line)
8   nomes_ig = re.search(r"(?<=\d{4}-\d{2}-\d{2},)([A-Z][a-z]+),([A-Z][a-z]+),",line)
9
10  if generos_ig and nomes_ig and idades:
11    if (generos_ig.group(0),idades.group(0)) not in dictig:
12      dictig[(generos_ig.group(0),idades.group(0))] = [nomes_ig.group(1) + " " +
          nomes_ig.group(2)]
13    else:
14      dictig[(generos_ig.group(0),idades.group(0))].append(nomes_ig.group(1) + " " +
          nomes_ig.group(2))
15
16 # a partir do dicion rio escreve-se o c digo html em que listamos os atleta por
     idade e g nero
17
18 messageig+="<p><strong>Mais ou com 35 anos</strong></p><ul>"
19 for gen in ['M','F']:
20   messageig+= "<li><strong>" + gen + "</strong><ol>"
21   for atleta in list(dictig.items()):
22     if atleta[0][0] == gen and int(atleta[0][1]) >= 35:
23       atlord=sorted(atleta[1])
24       for atl in atlord:
25         messageig+= "<li>" + atl + "</li>"
26   messageig+="</ol></li>"
27
28 messageig+="</ul><p><strong>Menos de 35 anos</strong></p><ul>"
29 for gen in ['M','F']:
30   messageig+= "<li><strong>" + gen + "</strong><ol>"
31   for atleta in list(dictig.items()):
32     if atleta[0][0] == gen and int(atleta[0][1]) < 35:
33       atlord=sorted(atleta[1])
34       for atl in atlord:
35         messageig+= "<li>" + atl + "</li>"
36   messageig+="</ol></li>"
37
38
39 fdig.write(messageig)
```

Distribuição por morada

Expressões regulares utilizadas

- **(?<=,**$[MF]$**,)**$[A-Z][a-z]+$**(?=,)**
  - **(?<=,**$[MF]$**,)** → Faz correspondência a, se existir na posição anterior, um 'M' ou 'F' entre vírgulas;
  - $[A-Z][a-z]+$ → Faz correspondência a uma letra maiúscula seguida de uma ou mais letras
  - **(?=,)** → Faz correspondência a, se existir uma ',' na posição seguinte.

- **(?<=,**$[MF]$**,)(**$[A-Z][a-z]+$**),(**$[A-Z][\hat{},]+$**)**
  - **,(**$[A-Z][\hat{},]+$**)** → Faz correspondência a uma ',' e uma letra maiúscula seguida de qualquer caracter exceto uma ','.
- **(?<=**$[0-9]$**{4}-**$[0-9]$**{2}-**$[0-9]$**{2},)(**$[A-Z][a-z]+$**),(**$[A-Z][a-z]+$**),**
  - **(?<=**$[0-9]$**{4}-**$[0-9]$**{2}-**$[0-9]$**{2},)** → Faz correspondência a, se existir uma data na posição anterior.

Aqui iremos explicar o código por detrás da figura 4.4.

Primeiramente, as moradas encontradas no dataset (**(?<=,**$[MF]$**,)**$[A-Z][a-z]+$**(?=,)**) serão guardadas em duas listas. A lista **moradas**, é a lista das moradas em que 2 ou mais atletas pertencem. A lista **moradasnrep**, é a lista das moradas em que só existe um atleta.

Depois, moradas, modalidades e nomes, serão guardados em variáveis (**(?<=,**$[MF]$**,)(**$[A-Z][a-z]+$**),(**$[A-Z][\hat{},]+$**)** - usada para pesquisar moradas e modalidades; **(?<=**$[0-9]$**{4}-**$[0-9]$**{2}-**$[0-9]$**{2},)(**$[A-Z][a-z]+$**),(**$[A-Z][a-z]+$**),** - usada para pesquisar os nomes). Variáveis essas que pertencem à constituição do dicionário **nresidentes** (keys: morada; valores: (nome,modalidade)), que é ordenado por morada.

A **messagemorada** será responsável por acumular os dados do dicionário.

```
1
2 # faz-se uma lista de moradas com repeti   es e outra sem repeti   es
3
4 moradas = []
5 moradasnrep = []
6 for line in csvreader[1:]:
7   n = re.search(r"(?<=,[MF],)[A-Z][a-z]+(?=,)",line)
8   if n and n.group(0) not in moradas:
9     moradasnrep.append(n.group(0))
10  if n:
11    moradas.append(n.group(0))
12
13
14 # Guarda-se os dados sobre moradas num dicion rio , morada : (nome,modalidade)
15
16 nresidentes = {}
17
18 for line in csvreader:
19   moradaemodal = re.search(r"(?<=,[MF],)([A-Z][a-z]+),([A-Z][^,]+)",line)
20   nomes = re.search(r"(?<=\d{4}-\d{2}-\d{2},)([A-Z][a-z]+),([A-Z][a-z]+),",line)
21
22   if moradaemodal and nomes:
23     if moradaemodal.group(1) not in nresidentes:
24       nresidentes[moradaemodal.group(1)] = []
25     nresidentes[moradaemodal.group(1)].append((nomes.group(1) + " " + nomes.group(2),
         moradaemodal.group(2)))
26
27 nresidentes = sorted(list(nresidentes.items()), key= lambda x : x[0])
28
29 # a partir do dicion rio escreve-se o c digo html em que listamos os atleta por
     morada
```

```
30
31 for localidade in nresidentes:
32   messagemorada+="<li><p><strong>" + localidade[0] + "</strong></p><ul>"
33   for residente in localidade[1]:
34     messagemorada+="<li>Nome: " + residente[0] + ",␣Modalidade: " +
         residente[1] + "</li>"
35   messagemorada+="</ul>"
36 messagemorada+="</ol>"
37
38 fdm.write(messagemorada)
```

Percentagem de Aptabilidade

Expressões regulares utilizadas

- $[0-9]$**{4}(?=(-**$[0-9]$**{2}-**$[0-9]$**{2}))** (já usada nas datas extremas);
- **(false|true),(false|true)** → Corresponde a dois valores de validade divididos por uma ',';

Passaremos a explicar o código responsável pelo desenvolvimento da imagem 4.5.

Começando pelo ciclo for que percorre o array **anos** (inicializado no início do exercício). Lendo o dataset, procura-se pelo ano em que o atleta aderiu ($[0-9]${4}(?=(-$[0-9]${2}-$[0-9]${2}))), e se é federado ou se tem resultado válido ((**false|true),(false|true)**). Se a pesquisa for bem sucedida, será guardado na lista (de strings) **laux** a aptabilidade do atleta. Para terminar, **messageapt** que é responsável por acumular todos os dados, também vai conter nela o resultado do cálculo das percentagens.

```
1
2
3 for ano in anos:
4   messageapt+= "<p><strong>" + ano + "</strong></p><ul>"
5   laux = []
6   for line in csvreader:
7     n = re.search(r"\d{4}(?=(-\d{2}-\d{2}))",line)
8     n1 = re.search(r"(false|true),(false|true)",line)
9     if n and n1 and n.group(0) == ano:
10      laux.append(n1.group(2))
11  if len(laux) > 0:
12    messageapt += "<li><p>" + "<strong>aptos</strong>␣=␣" + str(round(laux.count("
        true")/len(laux)*100,1)) + "%</p></li>"
13    messageapt += "<li><p>" + "<strong>n&atilde;o␣aptos</strong>␣=␣" + str(round(laux
        .count("false")/len(laux)*100,1)) + "%</p></li></ul>"
14
15
16 fapt.write(messageapt)
17
18 message+="</ul>"
19
20 fhtml.write(message)
```

3.2 EX 5 - Ficheiros CSV com listas e funções de agregação

O nosso objetivo neste exercício é converter um ficheiro **CSV** para o formato **JSON**, onde a primeira linha do ficheiro **CSV** vai funcionar como o cabeçalho da tabela **JSON** que define o que representa cada coluna.

Para o desenvolvimento do programa nós resolvemos dividir a nossa resolução por etapas, sendo estas:

1. A leitura do ficheiro *CSV* e a conversão para *JSON* em casos onde os elementos das linhas são atribuidos 1 por 1 com o cabeçalho, não havendo, portanto, a existência de listas
2. Aumentar a complexidade do código anterior para este aceitar ficheiros *CSV* com listas cujo tamanho é fixo
3. Aumentar a complexidade do código anterior para este aceitar ficheiros *CSV* com listas cujo tamanho está dentro de certo intervalo
4. Aumentar a complexidade do código anterior para este aceitar ficheiros *CSV* cujas listas vêm com uma função associada que as vai transformar

3.2.1 Conversão de CSV para JSON

Neste primeiro código, apenas é aberto o ficheiro **CSV** usando o *csv.reader* da biblioteca csv de python, o tal é percorrido e colocado numa lista de dicionários onde os elementos do cabeçalho são as keys e o elemento associado a estas são os values. Depois essa lista é convertida para formato **json** usando o *json.dumps* da biblioteca json de python e o resultado é imprimido.

```
import csv, json

with open("C:\\Users\\letsd\\OneDrive\\ rea de Trabalho\\uni\\PLCTP\\file1.csv", 'r') as file:
    csvreader = list(csv.reader(file))  # - Leitura do ficheiro csv
    json_list = []
    header = csvreader[0]               # - E isolado o cabecalho
    del csvreader[0]
    for m,info in enumerate(csvreader): # - Comeca a iteracao onde e
        json_list.append({})            # percorrido o ficheiro todo e se associa os valores
        for n,head in enumerate(header): # do cabecalho com os das restantes linhas
            json_list[m][head] = info[n]

print(json.dumps(json_list, indent=4,ensure_ascii=False)) # - Imprimido em formato JSON
```

3.2.2 Listas com tamanho definido

Agora é acrescentada ao programa a capacidade de receber listas pelo ficheiro **CSV**. Para indicar que certa coluna corresponde a listas o cabeçalho tem que ter o nome dessa coluna seguido do numero de elementos da lista dentro de chavetas *({5}, por exemplo).* No código é então feita uma pesquisa usando *re.search* da biblioteca re de python em todos os elementos do cabeçalho enquanto era preenchida a lista de dicionários, onde é usada a expressão regular **(?<={)**$[0-9]+$**(?=})**. Caso seja encontrada uma sequência de carateres que satisfaça a expressão regular começa então um ciclo que recolhe todos os elementos da lista, voltando de seguida à execução normal do programa.

Expressões regulares utilizadas

- **(?<={)**$[0-9]+$**(?=})**
  - **(?<={)** → Faz correspondência se existir um '{' na posição anterior
  - $[0-9]+$ → Faz correspondência a 1 ou mais dígitos
  - **(?=})** → Faz correspondência se existir um '}' na posição seguinte
- **{**$[0-9]+$**}** → Faz correspondência a '{' seguido de 1 ou mais dígitos e seguido de '}'

```
import csv, json, re

with open("C:\\Users\\letsd\\OneDrive\\ rea _de_Trabalho\\uni\\PLCTP\\file1.csv", 'r') as file:
    csvreader = list(csv.reader(file))
    json_list = []
    nvezes = 0
    header = csvreader[0]
    del csvreader[0]

    for m,info in enumerate(csvreader):
        json_list.append({})
        n = 0
        i = 0

        while i < len(info):
            nvezes = re.search(r"(?<={)\d+(?=})",header[n]) # - E procurada uma
                                                            # correspondencia com
                                                            # a expressao regular.
            if nvezes:                                      # - Verifica se foi
                nvezes = int(nvezes.group(0))               # encontrada correspondencia.
                head = re.sub(r"{\d+}","",header[n])  # - Usa o re.sub para apagar os
                n+=1                                  # carateres a mais no nome da lista.
                list_aux = []

                while nvezes > 0:           # - Come a um ciclo que vai
                    list_aux.append(info[i])  # guardar os elementos da lista.
                    nvezes-=1
                    n+=1
                    i+=1
                json_list[m][head] = list_aux
            else:
                json_list[m][header[n]] = info[i]
                n+=1
                i+=1

    print(json.dumps(json_list, indent=4,ensure_ascii=False))
```

3.2.3 Listas com um intervalo de tamanhos

De seguida, foi melhorado o programa de modo a que agora as listas possam ter tamanho variável. Isto é indicado no input usando o indicador do tamanho das listas *({5}, por exemplo)* acrescentando entre os { } mais um número com virgula *({3,5}, por exemplo, indica que o tamanho da lista poderá variar entre 3 e 5).* Para isso mudamos a expressão regular do código anterior para abrangir os casos anteriores e estes casos novos, **(?<={)**$[0-9]+$**(?=})**|**(?<={)(**($[0-9]+$**),**($[0-9]+$**)(?=})** . Caso seja encontrada uma sequência de caracteres que satisfaça esta expressão regular, é usada a expressão regular $[0-9]+$**,**$[0-9]+$ para verificar se a lista encontrada tem tamanho variável ou não, caso não seja, segue o caminho que se tinha no código anterior, caso contrário, guarda o tamanho mínimo e máximo da lista em varáveis e começa um cilo while (1), e imediatamente começa outro ciclo while (2), aqui vão ser recolhidos os elementos da lista até esta ter o comprimento mínimo, depois sai do ciclo (2) e enquanto tiver mais elementos para meter na lista percorre

o ciclo (1), quando este acabar o programa volta à sua execução normal.

Expressões regulares utilizadas

- **{**$[0-9]+$ → Faz correspondência a { seguido de 1 ou mais dígitos
- **(?<={)**$[0-9]+$**(?=})|(?<={)**$(([0-9]+)$**,**$([0-9]+))$**(?=})** → Faz correspondência se for satisfeito (?<={)$[0-9]+$(?=}) **ou** (?<={)$(([0-9]+)$,$([0-9]+))$(?=})
  - **(?<={)**$[0-9]+$**(?=})** *(Explicado nas expressões regulares de listas de tamanho definido)*
  - **(?<={)**$(([0-9]+)$**,**$([0-9]+))$**(?=})**
    - **(?<={)** → Faz correspondência se existir um '{' na posição anterior
    - $(([0-9]+)$**,**$([0-9]+))$ → Faz correspondência a 1 ou mais dígitos seguidos de "," seguido de 1 ou mais dígitos
    - **(?=})** → Faz correspondência se existir um '}' na posição seguinte

```
1  import csv, json, re
2
3  with open("C:\\Users\\letsd\\OneDrive\\ rea _de_Trabalho\\uni\\PLCTP\\file2.csv", 'r'
       ) as file:
4    csvreader = list(csv.reader(file))
5    json_list = []
6    nvezes = 0
7    header = csvreader[0]
8    del csvreader[0]
9    item = 0
10
11   while item < len(header):
12
13   # Como csv.reader vai separar as palavras do ficheiro por virgulas, Notas{3,5}
         ficaria dividido entre o 3 e o 5, neste ciclo e corrigido esse problema
14
15     if re.search(r"{\d+",header[item]) and item < len(header)-1:
16       header[item] = header[item] + "," + header[item+1]
17       del header[item+1]
18     item+=1
19
20   for m,info in enumerate(csvreader):
21     n = 0
22     i = 0
23     json_list.append({})
24     while i < len(info):
25       if info[i] != "" or header[n] != "":
26         nvezes = re.search(r"(?<={)\d+(?=})|(?<={)((\d+),(\d+))(?=})",header[n])
27         if nvezes:
28         # Verifica se a lista tem tamanho variavel
29           if re.search(r"\d+,\d+",nvezes.group(0)):
30             nmin = int(nvezes.group(2)) # Guarda o numero minimo de elementos da
                   lista
31             nmax = int(nvezes.group(3)) # Guarda o numero maximo de elementos da
                   lista
```

```
32              nvezes_int = nmax
33              head = re.sub(r"{((\d+),(\d+))}","",header[n])
34              n+=1
35              list_aux = []
36              while nvezes_int > 0:  # Come a o cilo 1
37                while nvezes_int > nmax - nmin:  # Come a o cilo 2
38                  list_aux.append(info[i])
39                  nvezes_int-=1
40                  n+=1
41                  i+=1
42
43                # Se nao existirem mais elementos para colocar na lista o ciclo 1 acaba
44
45                if info[i] == "":
46                  break
47                else:
48                  list_aux.append(info[i])
49                  nvezes_int-=1
50                  n+=1
51                  i+=1
52              json_list[m][head] = list_aux
53            else:
54              nvezes_int = int(nvezes.group(0))
55              head = re.sub(r"{\d+}","",header[n])
56              n+=1
57              list_aux = []
58              while nvezes_int > 0:
59                list_aux.append(info[i])
60                nvezes_int-=1
61                n+=1
62                i+=1
63              json_list[m][head] = list_aux
64          else:
65            json_list[m][header[n]] = info[i]
66            n+=1
67            i+=1
68        else:
69          i+=1
70          n+=1
71
72    print(json.dumps(json_list , indent=4,ensure_ascii=False))
```

3.2.4 Funções de agregação

Para concluir as etapas do desenvolvimento do programa foi então adicionada a capacidade de com cada lista se colocar no input "::" seguidos de uma função que seria executada na lista *(Notas{3,5}::sum, por exemplo)*. Para isso acrescentamos uma procura por expressão regular depois de se encontrar uma lista com a seguinte expressão, **(?<=::)[a-z]+**. Depois forma-se a lista normalmente e é usado if/elsif com as funções disponíveis para executar a função encontrada com a expressão regular mencionada anteriormente, depois disto o programa volta à sua execução normal.

Expressões regulares utilizadas

- **(?<=::)[a-z]+**
  - **(?<=::)** → Faz correspondência se existir um *::* na posição anterior
  - **[a-z]+** → Faz correspondência a 1 ou mais letras entre *a* e *z* (ascii)
- **::[a-z]+** → Faz correspondência a *::* seguido de 1 ou mais letras entre *a* e *z* (ascii)

```
1  import csv, json, re
2
3  with open("C:\\Users\\letsd\\OneDrive\\ rea de Trabalho\\uni\\PLCTP\\file3.csv", '
       r') as file:
4      csvreader = list(csv.reader(file))
5      json_list = []
6      nvezes = 0
7      header = csvreader[0]
8      del csvreader[0]
9      item = 0
10
11     while item < len(header):
12         if re.search(r"{\d+",header[item]) and item < len(header)-1:
13             header[item] = header[item] + "," + header[item+1]
14             del header[item+1]
15         item+=1
16
17     for m,info in enumerate(csvreader):
18         n = 0
19         i = 0
20         json_list.append({})
21         while i < len(info):
22             func = ""
23             if info[i] != "" or header[n] != "":
24                 nvezes = re.search(r"(?<={)\d+(?=})|(?<={)((\d+),(\d+))(?=})",header[n])
25                 if nvezes:
26                     func = re.search(r"(?<=::)[a-z]+",header[n]) # Verifica se h uma
                           funo associada lista
27                     if func:
28                         func = func.group(0)
29                     if re.search(r"\d+,\d+",nvezes.group(0)):
30                         nmin = int(nvezes.group(2))
31                         nmax = int(nvezes.group(3))
32                         nvezes_int = nmax
33                         head = re.sub(r"{((\d+),(\d+))}","",header[n])
34                         head = re.sub(r"::[a-z]+","",head)
35                         n+=1
36                         list_aux = []
37                         while nvezes_int > 0:
38                             while nvezes_int > nmax - nmin:
39                                 list_aux.append(int(info[i]))
40                                 nvezes_int-=1
41                                 n+=1
42                                 i+=1
43                             if info[i] == "":
```

```
44                      break
45                    else:
46                      list_aux.append(int(info[i]))
47                      nvezes_int-=1
48                      n+=1
49                      i+=1
50
51                # Para listas de tamanho variavel
52                # Se houver uma funcao associada a lista e procurada a funcao correta e
                      executa-a na lista
53
54                if func == "sum":
55                  json_list[m][head + "_sum"] = sum(list_aux)
56                elif func == "media":
57                  json_list[m][head + "_media"] = sum(list_aux)/len(list_aux)
58                else:
59                  json_list[m][head] = list_aux
60              else:
61                nvezes_int = int(nvezes.group(0))
62                head = re.sub(r"{\d+}","",header[n])
63                n+=1
64                list_aux = []
65                while nvezes_int > 0:
66                  list_aux.append(int(info[i]))
67                  nvezes_int-=1
68                  n+=1
69                  i+=1
70
71                # Para listas de tamanho fixo
72                # Se houver uma funcao associada a lista e procurada a funcao correta e
                      executa-a na lista
73
74                if func == "sum":
75                  json_list[m][head + "_sum"] = sum(list_aux)
76                elif func == "media":
77                  json_list[m][head + "_media"] = sum(list_aux)/len(list_aux)
78                else:
79                  json_list[m][head] = list_aux
80            else:
81              json_list[m][header[n]] = info[i]
82              n+=1
83              i+=1
84          else:
85            i+=1
86            n+=1
87
88      print(json.dumps(json_list , indent=4,ensure_ascii=False))
```

3.2.5 Extra - Escolha do ficheiro e pesquisa

Depois de se ter o programa a trabalhar com todas as funcionalidades que se pretendia, adicionamos ainda a possibilidade do utilizador dizer que ficheiro quer converter no inicio do programa e, em vez de imprimir

o resultado, o programa transfere-o para um ficheiro **JSON**, para isto no início é pedido um input com a função *input* onde o utilizador vai escrever o nome do ficheiro de input,depois o programa abre esse ficheiro e processa-o até ao final da execução, onde vai usar esse input para criar um novo ficheiro mudando o final do nome de *csv* para *json*, onde vai escrever o conteúdo **JSON**.
Para além disso foi também adicionada a capacidade de depois da execução do programa se aceder a elementos especificos do ficheiro **JSON**, para isto, no final do programa criamos um while que abre um menu em texto e só para quando o utilizador selecionar a opção de sair, aí, caso seja escolhida a opção de pesquisar, o utilizador terá de escolher por qual elemento do header pesquisar e depois escrever por qual string específica procura e são imprimidos os elementos de **JSON** compatíveis.

```
1  import csv
2  import json
3  import re
4
5  filestr = input("Nome do ficheiro que pretende converter: ")
6
7  with open("C:\\Users\\letsd\\OneDrive\\ rea de Trabalho\\uni\\PLCTP\\" + filestr,
       'r') as file:
8    csvreader = list(csv.reader(file))
9    json_list = []
10   nvezes = 0
11   header = csvreader[0]
12   del csvreader[0]
13   item = 0
14
15   while item < len(header):
16     if re.search(r"{\d+",header[item]) and item < len(header)-1:
17       header[item] = header[item] + "," + header[item+1]
18       del header[item+1]
19     item+=1
20
21   for m,info in enumerate(csvreader):
22     n = 0;i = 0
23     json_list.append({})
24     while i < len(info):
25       func = ""
26       if info[i] != "" or header[n] != "":
27         nvezes = re.search(r"(?<={)\d+(?=})|(?<={)((\d+),(\d+))(?=})",header[n])
28         if nvezes:
29           func = re.search(r"(?<=::)[a-z]+",header[n])
30           if func:
31             func = func.group(0)
32           if re.search(r"\d+,\d+",nvezes.group(0)):
33             nmin = int(nvezes.group(2))
34             nmax = int(nvezes.group(3))
35             nvezes_int = nmax
36             head = re.sub(r"{((\d+),(\d+))}","",header[n])
37             head = re.sub(r"::[a-z]+","",head)
38             n+=1
39             list_aux = []
40             while nvezes_int > 0:
41               while nvezes_int > nmax - nmin:
42                 list_aux.append(int(info[i]))
```

```
43                  nvezes_int-=1;n+=1;i+=1
44                if info[i] == "":
45                  break
46                else:
47                  list_aux.append(int(info[i]))
48                  nvezes_int-=1;n+=1;i+=1
49              if func == "sum":
50                json_list[m][head + "_sum"] = sum(list_aux)
51              elif func == "media":
52                json_list[m][head + "_media"] = sum(list_aux)/len(list_aux)
53              else:
54                json_list[m][head] = list_aux
55            else:
56              nvezes_int = int(nvezes.group(0))
57              head = re.sub(r"{\d+}","",header[n])
58              n+=1
59              list_aux = []
60              while nvezes_int > 0:
61                list_aux.append(int(info[i]))
62                nvezes_int-=1;n+=1;i+=1
63              if func == "sum":
64                json_list[m][head + "_sum"] = sum(list_aux)
65              elif func == "media":
66                json_list[m][head + "_media"] = sum(list_aux)/len(list_aux)
67              else:
68                json_list[m][head] = list_aux
69          else:
70            json_list[m][header[n]] = info[i]
71            n+=1;i+=1
72        else:
73          i+=1;n+=1
74    output = json.dumps(json_list, indent=4,ensure_ascii=False)
75    opf = open(filestr[:-3] + "json", "x")
76    opf.write(output)
77    opf.close()
78    pesquisar = 1
79    while pesquisar:
80      pesquisar = int(input("0 - Sair do programa\n1 - Pesquisar no ficheiro JSON\n")
          )
81      if pesquisar:
82        pesquisastr = "Escolhe um elemento do header:\n"
83        for n in range(len(header)-1):
84          find = re.search(r"({\d+}|{\d+,\d+})(::[a-z]+)?",header[n])
85          if find:
86            del header[n]
87        for n,i in enumerate(header):
88          if i != "":
89            pesquisastr = pesquisastr + str(n) + " - " + i + "\n"
90        elemheader = input(pesquisastr)
91        eleminfo = input("Por o qu que se vai pesquisar?\n")
92        encontros = []
93        for i in json_list:
94          if i[header[int(elemheader)]] == eleminfo:
95            encontros.append(i)
```

```
96          print(json.dumps(encontros, indent=4, ensure_ascii=False))
```

Capítulo 4

Exemplos de utilização

4.1 EX 2 - Processador de Registos de Exames Médicos Desportivos

Ao executar o programa vamos obter as seguintes páginas **html**:

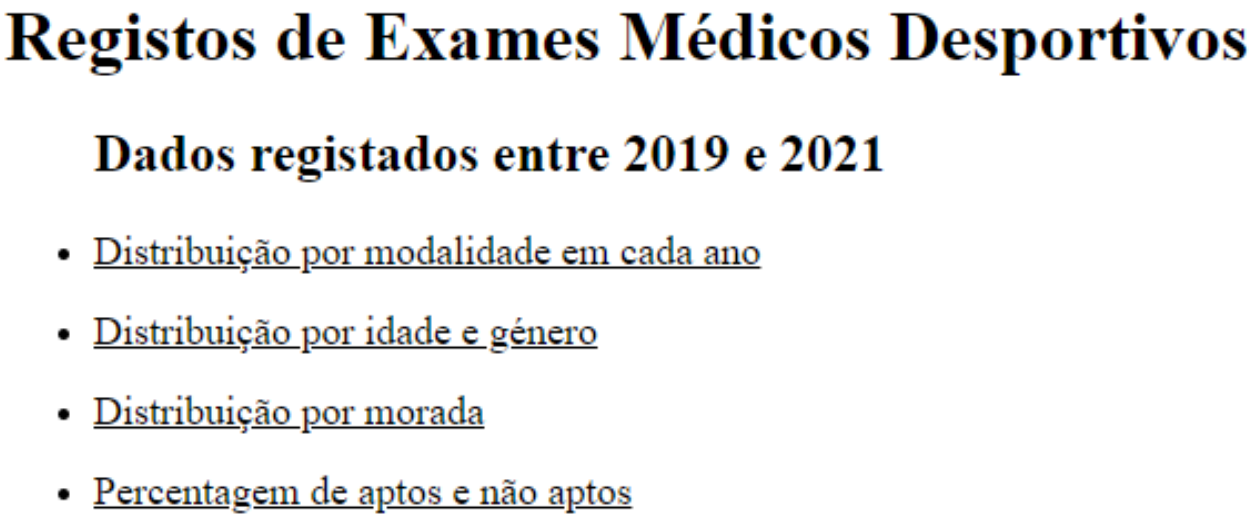

Figura 4.1: Página index.html

Distribuição por modalidade

	Andebol	Atletismo	BTT	Badminton	Basquetebol	Ciclismo	Dança	Equitação	Esgrima	Futebol	Karaté	Orientação	Parapente	Patinagem	Triatlo
2019:	11	9	12	12	9	11	11	7	5	13	9	8	10	9	9
2020:	9	6	16	10	12	12	10	5	9	7	11	8	5	4	14
2021:	1	1	1	2	3	0	1	1	1	1	1	1	2	1	0
Total:	21	16	29	24	24	23	22	13	15	21	21	17	17	14	23

2019

- **Andebol**
  1. Bell Zimmerman
  2. Blackburn Price
  3. Farrell Gilliam
  4. Flynn York
  5. Hood Burks
  6. Jacobs Carroll
  7. Liza Webb
  8. Tami Lawrence
  9. Tanya Ballard
  10. Villarreal Walton
  11. Waters Dale
- **Atletismo**
  1. Dana Morrison
  2. Maria Mcfadden
  3. Marla Kelley
  4. Meagan Le
  5. Myrna Guerrero
  6. Rebekah Mayer
  7. Sheppard Greer
  8. Staci Jarvis
  9. Vang George
- **BTT**
  1. Andrea Hensley
  2. Brooks Gamble
  3. Carmella Shaw
  4. Charles Bowman
  5. Kent Tyler
  6. Le Blair
  7. Munoz Meyers
  8. Nielsen Tanner
  9. Powers Torres
  10. Rhea Patel
  11. Robbie Bailey

Figura 4.2: Página DistribModalidade.html

Distribuição por idade e género

	M	F
Menos de 35 anos:	134	152
Mais ou com 35 anos:	7	7

Mais ou com 35 anos

- **M**
  1. Anita Lyons
  2. Dana Morrison
  3. Hardin Mejia
  4. Hodge Maxwell
  5. Kim Combs
  6. Miller Rojas
  7. Price Hardin
- **F**
  1. Beck Stevenson
  2. Bowers Gilliam
  3. Frank Bishop
  4. Lucia Bright
  5. Marsh Crosby
  6. Tanisha Castro
  7. Vang George

Menos de 35 anos

- **M**
  1. Albert Sweeney
  2. Banks Guerra
  3. Foreman Prince
  4. Jennie Hawkins
  5. Liza Webb
  6. Merrill Maddox
  7. Staci Jarvis
  8. Stefanie Byrd
  9. Welch Rosales
  10. Cheryl Berger
  11. Hollie Solis
  12. Kristen Jacobs
  13. Lester Strong
  14. Mcdonald Bender
  15. Pacheco Robbins
  16. Schmidt Hopper
  17. Trujillo Burke

Figura 4.3: Página DistribIG.html

Distribuição por morada

1. **Aguila**
   - Nome: Schmidt Hopper, Modalidade: Basquetebol
2. **Alafaya**
   - Nome: Carmella Lindsay, Modalidade: Futebol
3. **Alamo**
   - Nome: Lilian Levine, Modalidade: Parapente
4. **Alden**
   - Nome: Young Estes, Modalidade: Ciclismo
5. **Alfarata**
   - Nome: Elisa Bernard, Modalidade: Futebol
6. **Allamuchy**
   - Nome: Frieda Hansen, Modalidade: Ciclismo
7. **Alleghenyville**
   - Nome: Julianne Powell, Modalidade: Andebol
8. **Allentown**
   - Nome: Tania Sargent, Modalidade: Basquetebol
9. **Aurora**
   - Nome: Shelton Ingram, Modalidade: Patinagem
10. **Austinburg**
    - Nome: Alyssa Melendez, Modalidade: Dança
11. **Axis**
    - Nome: Nola Wade, Modalidade: Karaté
12. **Babb**
    - Nome: Graves Goff, Modalidade: Andebol
    - Nome: Joann Barnes, Modalidade: Basquetebol
13. **Bagtown**

Figura 4.4: Página DistribMorada.html

Percentagem de aptos e não aptos

2019

- **aptos** = 53.8%
- **não aptos** = 46.2%

2020

- **aptos** = 54.3%
- **não aptos** = 45.7%

2021

- **aptos** = 52.9%
- **não aptos** = 47.1%

Figura 4.5: Página Aptidao.html

4.2 EX 5 - Ficheiros CSV com listas e funções de agregação

Mostram-se a seguir alguns testes feitos (valores introduzidos) e os respectivos resultados obtidos:

4.2.1 Exemplo 1

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

4.2.2 Exemplo 2

```
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
```

4.2.3 Exemplo 3

```
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,
```

4.2.4 Exemplo 4

```
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,
```

4.2.5 Exemplo Extra

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

4.2.6 Outputs

{} file1.json > ...

```
[
  {
    "Número": "264",
    "Nome": "Marcelo Sousa",
    "Curso": "Ciência Política"
  },
  {
    "Número": "3162",
    "Nome": "Cândido Faísca",
    "Curso": "Teatro"
  },
  {
    "Número": "7777",
    "Nome": "Cristiano Ronaldo",
    "Curso": "Desporto"
  }
]
```

Figura 4.6: Exemplo1 - Ficheiro JSON

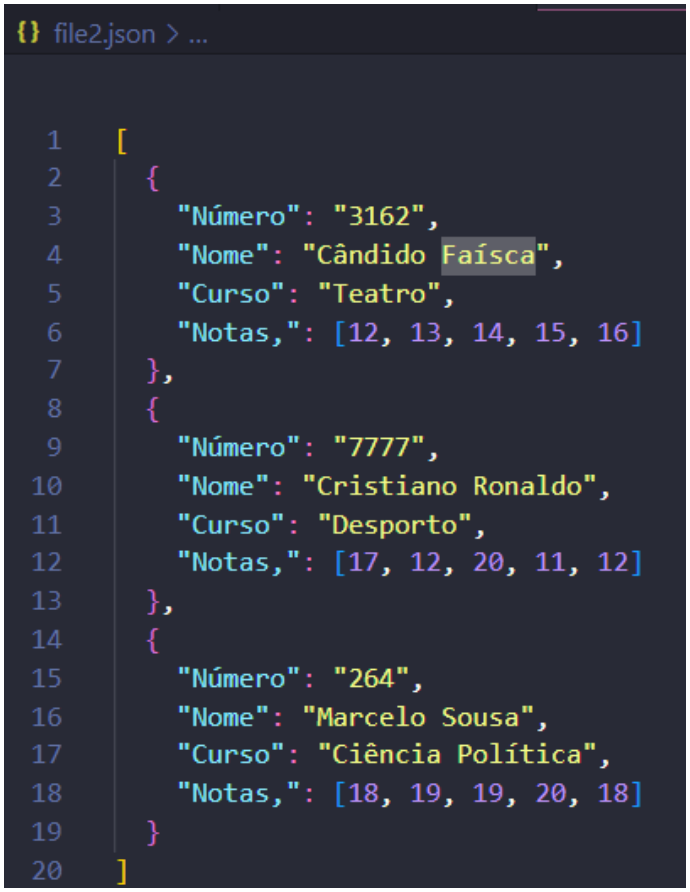

Figura 4.7: Exemplo2 - Ficheiro JSON

file3.json > ...

```json
[
  {
    "Número": "3162",
    "Nome": "Cândido Faísca",
    "Curso": "Teatro",
    "Notas": [12, 13, 14]
  },
  {
    "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]
  }
]
```

Figura 4.8: Exemplo3 - Ficheiro JSON

```
[
    {
        "Número": "3162",
        "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
    }
]
```


Figura 4.9: Exemplo4 - Ficheiro JSON

```
C:\Users\letsd\OneDrive\Área de Trabalho\uni\PLCTP>python ex5.py
Nome do ficheiro que pretende converter: file1.csv
0 - Sair do programa
1 - Pesquisar no ficheiro JSON
1
Escolhe um elemento do header:
0 - NÃºmero
1 - Nome
2 - Curso
1
Por o quê que se vai pesquisar?
Cristiano Ronaldo
[
    {
        "NÃºmero": "7777",
        "Nome": "Cristiano Ronaldo",
        "Curso": "Desporto"
    }
]
0 - Sair do programa
1 - Pesquisar no ficheiro JSON
```

Figura 4.10: ExemploExtra - CMD

Capítulo 5

Conclusão

Com a concretização deste trabalho alcançamos os objetivos esperados, o nosso grupo é agora mais capaz na manipulação de expressões regulares e na sua aplicação em situações reais.
Para além disso, com este trabalho aprendemos a escrever código HTML e CSS e também como funcionam ficheiros CSV e JSON.
Finalmente, este trabalho fortificou imenso as nossas bases, dando-nos a possibilidade de tirar o melhor proveito possível da Unidade Curricular.

Apêndice A

Código dos Programas

A.1 Processador de Registos de Exames Médicos Desportivos

```
1
2 import re, os
3
4 # Abrir ficheiros ----------------------
5 try:
6   fhtml = open("index.html","x")
7 except FileExistsError:
8   os.remove("index.html")
9   fhtml = open("index.html","x")
10
11 f = open("C:\\Users\\letsd\\OneDrive\\ rea de Trabalho\\uni\\PLCTP\\emd.csv", 'r')
12
13
14 # Separar as linhas de emd em lista de strings
15
16 csvreader = f.readlines()
17
18 # message vai guardar todo o c digo html de index.html
19 message = "<html><head><style>a{font-size: 18px; color: black;}h1{font-size: 32px;}h2{
     font-size: 24px;}li{margin-bottom: 15px;}</style></head><body><h1>Registos de
     Exames M&eacute;dicos Desportivos</h1><ul>"
20
21 # Datas extremas dos registos no dataset ------------------------------
22
23 anos = set()
24 for line in csvreader:
25   n = re.search(r"\d{4}(?=(-\d{2}-\d{2}))",line)
26   if n:
27     anos.add(n.group(0))
28 anos = sorted(list(anos))
29
30 message += "<h2>Dados registados entre " + anos[0] + " e " + anos[-1] + "</h2>"
31
```

```
# Distribuição por modalidade em cada ano e no total
-----------------------------------------------------

try:
  fdmod = open("DistribModalidade.html","x")
except FileExistsError:
  os.remove("DistribModalidade.html")
  fdmod = open("DistribModalidade.html","x")

messagemodalidade = "<html><head></head><body><h1>Distribui&ccedil;&atilde;o por modalidade</h1><ol>"

message+= "<li><a href=\"DistribModalidade.html\">Distribui&ccedil;&atilde;o por modalidade em cada ano</a></li>"

modalidades = set()
for line in csvreader[1:]:
  n = re.search(r"(?<=,[MF],)([A-Z][a-z]+),([A-Z][^,]+)",line)
  if n:
    modalidades.add(n.group(2))

modalidades = sorted(list(modalidades))

messagemodalidade+="<div class=\"container\">"

messagemodalidade += "<div class=\"sub-container\">"
for ano in anos:
  messagemodalidade+= "<p><strong>" + ano + ":</strong>" + "</p>"
messagemodalidade+= "<p><strong>Total:</strong></p></div>"


for i in modalidades:
  messagemodalidade += "<div class=\"sub-container\"><head><strong>" + i + "</strong></head>"
  total = 0
  for ano in anos:
    counter = 0
    messagemodalidade+="<p>"
    for line in csvreader:
      nanos = re.search(r"\d{4}(?=(-\d{2}-\d{2}))",line)
      nmodalidade = re.search(r"(?<=,[MF],)([A-Z][a-z]+),([A-Z][^,]+)",line)
      if nanos and nmodalidade and nanos.group(0) == ano and nmodalidade.group(2) == i:
        counter+=1
    total+=counter
    messagemodalidade+= str(counter) + "</p>"
  messagemodalidade+= "<p>" + str(total) + "</p></div>"
messagemodalidade+="</div>"

messagemodalidade = messagemodalidade[:60] + "<style>h1{font-size: 32px;}.container{display: flex;gap: 10px;align-items: flex-end;}</style>" + messagemodalidade[60:]

messagemodalidade+="<p>----------</p>"

```

```
# Guarda-se os dados sobre modalidades num dicion rio , (modalidade, ano) : nome

nmod = {}

for line in csvreader:
  moradaemodal_dm = re.search(r"(?<=,[MF],)([A-Z][a-z]+),([A-Z][^,]+)",line)
  anos_dm = re.search(r"\d{4}(?=(-\d{2}-\d{2}))",line)
  nomes_dm = re.search(r"(?<=\d{4}-\d{2}-\d{2},)([A-Z][a-z]+),([A-Z][a-z]+),",line)

  if moradaemodal_dm and nomes_dm and anos_dm:
    if (moradaemodal_dm.group(2),anos_dm.group(0)) not in nmod:
      nmod[(moradaemodal_dm.group(2),anos_dm.group(0))] = [nomes_dm.group(1) + " " +
          nomes_dm.group(2)]
    else:
      nmod[(moradaemodal_dm.group(2),anos_dm.group(0))].append(nomes_dm.group(1) + "
          " + nomes_dm.group(2))

# modalidades j  definida
modalidadesord = sorted(modalidades)
# anos j  definida nas datas extremas

# a partir do dicion rio escreve-se o c digo html em que listamos os atleta por
    modalidade e ano

for ano in anos:
  messagemodalidade += "<p><strong>" + ano + "</strong></p><ul>"
  for mod in modalidadesord:
    messagemodalidade += "<li><strong>" + mod + "</strong><ol>"
    for atletas in list(nmod.items()):
      if atletas[0][1] == ano and atletas[0][0] == mod:
        atlord = sorted(atletas[1])
        for atleta in atlord:
          messagemodalidade += "<li>" + atleta + "</li>"
    messagemodalidade+= "</ol></li>"
  messagemodalidade+="</ul>"


fdmod.write(messagemodalidade)

#  Distribui  o por idade e g nero
    ---------------------------------------------------------------------

try:
  fdig = open("DistribIG.html","x")
except FileExistsError:
  os.remove("DistribIG.html")
  fdig = open("DistribIG.html","x")

messageig = "<html><head></head><body><h1>Distribui&ccedil;&atilde;o por idade e g&
    eacute;nero</h1><ol>"

message+= "<li><a href = \"DistribIG.html\">Distribui&ccedil;&atilde;o por idade e g&
    eacute;nero</a></li>"
```

```
messageig+= "<div class=\"container\"><div class=\"sub-container\"><p><strong>Menos de 35 anos:</strong></p>"
messageig+="<p><strong>Mais ou com 35 anos:</strong></p></div>"


# Constru o da tabela com a distribui o

for i in ['M','F']:
  countmenor=0
  countmaior=0
  messageig+= "<div class=\"sub-container\"><p>"
  for line in csvreader:
    genero = re.search(r"(?<=,)"+i+r"(?=,)",line)
    idade = re.search(r"(?<=,)\d{2}(?=,[MF])",line)
    if genero and idade and genero.group(0) == i and int(idade.group(0)) < 35:
      countmenor +=1
    elif genero and idade and genero.group(0) == i and int(idade.group(0)) >= 35:
      countmaior +=1
  messageig+= "<strong>" + i + "</strong>" + "</p><p>" + str(countmenor) + "</p><p>" + str(countmaior) + "</p></div>"

messageig+="</div>"

messageig = messageig[:60] + "<style>h1{font-size: 32px;}.container{display:flex;gap: 10px;align-items: flex-end;}</style>" + messageig[60:]

messageig+="<p>------------------------------------------------------------------------</p>"

# Guarda-se os dados sobre idade e g nero num dicion rio, (genero, idade) : nome

dictig = {}

for line in csvreader:
  generos_ig = re.search(r"(?<=,)[MF](?=,)",line)
  idades = re.search(r"(?<=,)\d{2}(?=,[MF])",line)
  nomes_ig = re.search(r"(?<=\d{4}-\d{2}-\d{2},)([A-Z][a-z]+),([A-Z][a-z]+),",line)

  if generos_ig and nomes_ig and idades:
    if (generos_ig.group(0),idades.group(0)) not in dictig:
      dictig[(generos_ig.group(0),idades.group(0))] = [nomes_ig.group(1) + " " + nomes_ig.group(2)]
    else:
      dictig[(generos_ig.group(0),idades.group(0))].append(nomes_ig.group(1) + " " + nomes_ig.group(2))

# a partir do dicion rio escreve-se o c digo html em que listamos os atleta por idade e g nero

messageig+="<p><strong>Mais ou com 35 anos</strong></p><ul>"
for gen in ['M','F']:
  messageig+= "<li><strong>" + gen + "</strong><ol>"
  for atleta in list(dictig.items()):
    if atleta[0][0] == gen and int(atleta[0][1]) >= 35:
      atlord=sorted(atleta[1])
```

```
175         for atl in atlord:
176           messageig+= "<li>" + atl + "</li>"
177     messageig+="</ol></li>"
178 
179 messageig+="</ul><p><strong>Menos de 35 anos</strong></p><ul>"
180 for gen in ['M','F']:
181     messageig+= "<li><strong>" + gen + "</strong><ol>"
182     for atleta in list(dictig.items()):
183       if atleta[0][0] == gen and int(atleta[0][1]) < 35:
184         atlord=sorted(atleta[1])
185         for atl in atlord:
186           messageig+= "<li>" + atl + "</li>"
187     messageig+="</ol></li>"
188 
189 
190 fdig.write(messageig)
191 
192 # Distribui  o por morada
    ------------------------------------------------------------------------
193 
194 message+="<li><a href=\"DistribMorada.html\">Distribui&ccedil;&atilde;o por morada</a
      ></li>"
195 try:
196     fdm = open("DistribMorada.html","x")
197 except FileExistsError:
198     os.remove("DistribMorada.html")
199     fdm = open("DistribMorada.html","x")
200 
201 messagemorada = "<html><head><style>h1{font-size: 32px;}</style></head><body><h1>
      Distribui&ccedil;&atilde;o por morada</h1><ol>"
202 
203 # faz-se uma lista de moradas com repeti  es e outra sem repeti  es
204 
205 moradas = []
206 moradasnrep = []
207 for line in csvreader[1:]:
208     n = re.search(r"(?<=,[MF],)[A-Z][a-z]+(?=,)",line)
209     if n and n.group(0) not in moradas:
210       moradasnrep.append(n.group(0))
211     if n:
212       moradas.append(n.group(0))
213 
214 
215 # Guarda-se os dados sobre moradas num dicion rio, morada : (nome,modalidade)
216 
217 nresidentes = {}
218 
219 for line in csvreader:
220     moradaemodal = re.search(r"(?<=,[MF],)([A-Z][a-z]+),([A-Z][^,]+)",line)
221     nomes = re.search(r"(?<=\d{4}-\d{2}-\d{2},)([A-Z][a-z]+),([A-Z][a-z]+),",line)
222 
223     if moradaemodal and nomes:
224       if moradaemodal.group(1) not in nresidentes:
225         nresidentes[moradaemodal.group(1)] = []
```

```
226     nresidentes[moradaemodal.group(1)].append((nomes.group(1) + " " + nomes.group(2),
            moradaemodal.group(2)))
227
228 nresidentes = sorted(list(nresidentes.items()), key= lambda x : x[0])
229
230 # a partir do dicion rio escreve-se o c digo html em que listamos os atleta por
        morada
231
232 for localidade in nresidentes:
233   messagemorada+="<li><p><strong>" + localidade[0] + "</strong></p><ul>"
234   for residente in localidade[1]:
235     messagemorada+="<li>Nome: " + residente[0] + ", Modalidade: " +
            residente[1] + "</li>"
236   messagemorada+="</ul>"
237 messagemorada+="</ol>"
238
239 fdm.write(messagemorada)
240
241 # Percentagem de aptos e n o aptos por ano
    ----------------------------------------------------------------------
242
243 message+="<li><a href=\"Aptidao.html\">Percentagem de aptos e n&atilde;o aptos</a></
        li>"
244 try:
245   fapt = open("Aptidao.html","x")
246 except FileExistsError:
247   os.remove("Aptidao.html")
248   fapt = open("Aptidao.html","x")
249
250 messageapt = "<html><head><style>h1{font-size: 32px;}</style></head><body><h1>
        Percentagem de aptos e n&atilde;o aptos</h1><ol>"
251
252 for ano in anos:
253   messageapt+= "<p><strong>" + ano + "</strong></p><ul>"
254   laux = []
255   for line in csvreader:
256     n = re.search(r"\d{4}(?=(-\d{2}-\d{2}))",line)
257     n1 = re.search(r"(false|true),(false|true)",line)
258     if n and n1 and n.group(0) == ano:
259       laux.append(n1.group(2))
260   if len(laux) > 0:
261     messageapt += "<li><p>" + "<strong>aptos</strong> = " + str(round(laux.count("
            true")/len(laux)*100,1)) + "%</p></li>"
262     messageapt += "<li><p>" + "<strong>n&atilde;o aptos</strong> = " + str(round(laux
            .count("false")/len(laux)*100,1)) + "%</p></li></ul>"
263
264
265 fapt.write(messageapt)
266
267
268
269 message+="</ul>"
270
271 fhtml.write(message)
```

```
272 fhtml.close()
273 fdmod.close()
274 fapt.close()
275 fdm.close()
276 f.close()
```

A.2 Ficheiros CSV com listas e funções de agregação

```
1  import csv
2  import json
3  import re
4
5  filestr = input("Nome do ficheiro que pretende converter: ")
6
7  with open("C:\\Users\\letsd\\OneDrive\\ rea de Trabalho\\uni\\PLCTP\\" + filestr, 'r
       ') as file:
8    csvreader = list(csv.reader(file))
9    json_list = []
10   nvezes = 0
11   header = csvreader[0]
12   del csvreader[0]
13   item = 0
14
15   while item < len(header):
16     if re.search(r"{\d+",header[item]) and item < len(header)-1:
17       header[item] = header[item] + "," + header[item+1]
18       del header[item+1]
19     item+=1
20
21   for m,info in enumerate(csvreader):
22     n = 0;i = 0
23     json_list.append({})
24     while i < len(info):
25       func = ""
26       if info[i] != "" or header[n] != "":
27         nvezes = re.search(r"(?<={)\d+(?=})|(?<={}((\d+),(\d+))(?=})",header[n])
28         if nvezes:
29           func = re.search(r"(?<=::)[a-z]+",header[n])
30           if func:
31             func = func.group(0)
32           if re.search(r"\d+,\d+",nvezes.group(0)):
33             nmin = int(nvezes.group(2))
34             nmax = int(nvezes.group(3))
35             nvezes_int = nmax
36             head = re.sub(r"{((\d+),(\d+))}","",header[n])
37             head = re.sub(r"::[a-z]+","",head)
38             n+=1
39             list_aux = []
40             while nvezes_int > 0:
41               while nvezes_int > nmax - nmin:
42                 list_aux.append(int(info[i]))
```

```
43                  nvezes_int-=1;n+=1;i+=1
44                if info[i] == "":
45                  break
46                else:
47                  list_aux.append(int(info[i]))
48                  nvezes_int-=1;n+=1;i+=1
49              if func == "sum":
50                json_list[m][head + "_sum"] = sum(list_aux)
51              elif func == "media":
52                json_list[m][head + "_media"] = sum(list_aux)/len(list_aux)
53              else:
54                json_list[m][head] = list_aux
55            else:
56              nvezes_int = int(nvezes.group(0))
57              head = re.sub(r"{\d+}","",header[n])
58              n+=1
59              list_aux = []
60              while nvezes_int > 0:
61                list_aux.append(int(info[i]))
62                nvezes_int-=1;n+=1;i+=1
63              if func == "sum":
64                json_list[m][head + "_sum"] = sum(list_aux)
65              elif func == "media":
66                json_list[m][head + "_media"] = sum(list_aux)/len(list_aux)
67              else:
68                json_list[m][head] = list_aux
69          else:
70            json_list[m][header[n]] = info[i]
71            n+=1;i+=1
72        else:
73          i+=1;n+=1
74    output = json.dumps(json_list, indent=4,ensure_ascii=False)
75    opf = open(filestr[:-3] + "json", "x")
76    opf.write(output)
77    opf.close()
78    pesquisar = 1
79    while pesquisar:
80      pesquisar = int(input("0 - Sair do programa\n1 - Pesquisar no ficheiro JSON\n"))
81      if pesquisar:
82        pesquisastr = "Escolhe um elemento do header:\n"
83        for n in range(len(header)-1):
84          find = re.search(r"({\d+}|{\d+,\d+})(::[a-z]+)?",header[n])
85          if find:
86            del header[n]
87        for n,i in enumerate(header):
88          if i != "":
89            pesquisastr = pesquisastr + str(n) + " - " + i + "\n"
90        elemheader = input(pesquisastr)
91        eleminfo = input("Por o qu que se vai pesquisar?\n")
92        encontros = []
93        for i in json_list:
94          if i[header[int(elemheader)]] == eleminfo:
95            encontros.append(i)
96        print(json.dumps(encontros,indent=4,ensure_ascii=False))
```