

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

Trabalho Prático 1

Relatório de Desenvolvimento

Breno Fernando Guerra Marrão
A97768

Tales André Rovaris Machado
A96314

Tiago Passos Rodrigues
A96414

9 de novembro de 2022

Conteúdo

1	Introdução	2
2	Exercício 1	3
2.1	Análise e concepção da resolução dos problemas	3
2.1.1	Alínea a)	3
2.1.2	Alínea b)	3
2.1.3	Alínea c)	4
2.1.4	Alínea d)	4
2.2	Testes	5
2.2.1	Alínea a)	5
2.2.2	Alínea b)	6
2.2.3	Alínea c)	7
2.2.4	Alínea d)	7
3	Exercício 5	8
3.1	Análise e concepção da resolução do problema	8
3.2	Testes	9
3.2.1	alunos.csv	9
3.2.2	alunos2.csv	10
3.2.3	alunos3.csv	12
3.2.4	alunos4.csv	14
3.2.5	alunos5.csv	16
4	Conclusão	18
A	Código do Programa	19
A.1	Exercício 1	19
A.1.1	Alínea a)	19
A.1.2	Alínea b)	20
A.1.3	Alínea c)	21
A.1.4	Alínea d)	22
A.2	Exercício 2	23

Capítulo 1

Introdução

Este relatório no âmbito de processamento de linguagens e compiladores, teve como o objetivo a análise e resolução dos problemas propostos, através dos conhecimentos e estudos de expressões regulares e com o uso dos módulos re de expressões regulares do python. Foram escolhidos 2 exercícios sendo eles o exercício 1 e 5, sendo o primeiro a análise, recolha de dados sobre processos, tratamento e conversão de dados para json e o segundo a conversão entre ficheiros csv para o formato json.

Estrutura do Relatório

O relatório está dividido em duas grandes partes, o primeiro e quinto problema, e em cada um deles existem 2 subcapítulos, que são respectivamente: Análise e concepção da resolução dos problemas e Testes. Temos também um capítulo final onde falamos sobre as conclusões tiradas com o projeto.

Capítulo 2

Exercício 1

2.1 Análise e concepção da resolução dos problemas

Ao analisarmos o arquivo "processos.txt" reparamos que existem clones de certas linhas. Como está ordenado por ordem alfabética esses clones estão seguidos uns dos outros, portanto, resolvemos guardar o anterior e comparar com a linha que está a ser analisada e só tratamos a informação caso seja diferente.

2.1.1 Alínea a)

Na primeira alínea do exercício 1 é nos pedido que calculemos a frequência de processos por ano. Decidimos que a melhor maneira de tratar essa informação seria um dicionário em que a chave é o ano. Percorremos o arquivo "processos.txt" linha por linha, por cada linha única aplicamos a função `match` do módulo `re` pois só precisamos analisar a parte inicial da frase com a seguinte expressão regular:

```
r'([1-9][0-9]*)::([0-9]{4})'
```

no qual temos um grupo que diz que temos que ter um número de 1 a 9 podendo ser seguidos por outros números de 0 a 9 e depois separado por ":" temos outro grupo que será o ano.

Caso de facto exista na frase analisada esse padrão no início da linha iremos adicionar ao dicionário na chave que seja o ano reconhecido.

2.1.2 Alínea b)

Na segunda alínea do exercício 1 é pedido que separemos os nomes e apelidos das pessoas presentes agrupados em séculos. Para resolver esse problema criamos 2 dicionários nos quais um deles é a contagem de cada nome que aparece naquele século, sendo o século a chave do dicionário, e no segundo os apelidos, também no mesmo século.

Para resolver então analisamos de linha a linha e primeiramente fazemos o `split` pelo separador do documento, nesse caso ":", e a partir de então com os diferentes elementos do resultado fazemos a separação e armazenamento dos dados.

Primeiro verificamos em qual século o processo se encontra e verificamos se a chave do devido século existe. Logo em seguida separamos os diferentes nomes e adicionamos ao dicionário através da função auxiliar `add_dic(pessoa)` que recebe um nome e separa o primeiro e último nome e adiciona no dicionário dos nomes e apelidos no devido século especificado. Após o processo acabar temos um pequeno bloco de código para a visualização e apresentação dos dados.

2.1.3 Alínea c)

Para a resolução da terceira alínea fazemos o mesmo processo de análise por linhas e também armazenamos o resultado em um dicionário chamado parentesco em que cada chave é o nome do parentesco. Primeiro, semelhante ao segundo problema, fazemos o split de cada linha pelo separador ":" e então procuramos pela expressão regular:

```
r'\,((([A-Z][a-z]+[ ]*)+)\.[ ]*Proc'
```

que representa o parentesco, que esta sempre seguido do sufixo Proc, cujo indentifica que realmente é um parentesco. Após isso verificamos se o parentesco já esta no dicionário e se não, adicionamos, caso contrário, fazemos a contagem. Como foi feito nas outras alíneas, fizemos um pequeno bloco de código para visualização dos dados.

2.1.4 Alínea d)

Na última alínea do exercício 1 é proposto transformamos os 20 primeiros registos num novo ficheiro em formato json. Para resolver este problema vamos usar dois módulos, o json e o re.

Escolhemos um dicionário em que o número de cada linha serão as chaves do dicionário e dentro de cada chave está outro dicionário na qual terá como chave o número do processo, o ano e as pessoas envolvidas que está representado como uma lista de dicionários para cada pessoa em que são opcionais o n^o.processo e o parentesco.

Para resolver então analisamos de linha a linha e primeiramente fazemos o split pelo separador do documento, nesse caso ":", separamos o primeiro elemento desse split pra ser o número e o segundo elemento para ser o ano.

Após isso percorremos do segundo elemento até o penúltimo elemento onde aplicamos a função findall do modulo re com a expressão regular:

```
r'([a-zA-Z ]+)(\,[a-zA-Z ]+)(\.[ ]*Proc\.[0-9]+)'
```

utilizamos a função findall pois podemos ter mais do que uma pessoa entre o separador ":". Percorremos a lista resultante do findall ,caso não tenha resultado , sabemos que só tem o nome da pessoa e fazemos append à lista de pessoas, caso contrário sabemos que temos a informação do parentesco e do n^o.processo da pessoa e adicionamos isso ao dicionario da pessoa e fazemos append.

2.2 Testes

2.2.1 Alínea a)

```
○ → Ex1 git:(main) python3 frequencia_ano.py
Qual o ano? 1915
Não existe processo
Qual o ano? 1912
Não existe processo
Qual o ano?1910
Ano: 1910 Resultado: 27
Qual o ano?1745
Ano: 1745 Resultado: 44
Qual o ano?1880
Ano: 1880 Resultado: 59
Qual o ano?1650
Ano: 1650 Resultado: 1
Qual o ano?1649
Não existe processo
Qual o ano?1640
Não existe processo
Qual o ano?
```

2.2.2 Alínea b)

```
○ → Ex1 git:(main) X python3 nomes.py
Qual o seculo? 17
Qual o nome? Joao
Qual o apelido? Silva
Seculo: 17 Nome: 1291 Apelido: 421
Qual o seculo? 18
Qual o nome? Fernando
Qual o apelido? Rodrigues
Seculo: 18 Nome: 140 Apelido: 2377
Qual o seculo? 19
Qual o nome? Hugo
Qual o apelido? Ribeiro
Não existe nome ou apelido neste século
Qual o seculo? 19
Qual o nome? Manuel
Qual o apelido? Sousa
Seculo: 19 Nome: 4035 Apelido: 967
Qual o seculo? █
```

2.2.3 Alínea c)

```
Ex1 git:(main) X python3 frequenciapar.py
{'Tio Paterno': 1853, 'Tio Materno': 1931, 'Irmão': 12350, 'Primo Paterno': 160, 'Sobrinho Materno': 1692, 'Pai': 468, 'Filho': 345, 'Sobrinho Patern
o': 1641, 'Irmãos': 686, 'Sobrinhos Maternos': 98, 'Irmão Paterno': 487, 'Neto Materno': 41, 'Sobrinhos Paternos': 57, 'Sobrinho Neto Paterno': 95, '
Primo': 638, 'Primo Materno': 225, 'Tio Avo Paterno': 98, 'Irmão Materno': 44, 'Sobrinho Bisneto Paterno': 3, 'Tios Maternos': 20, 'Irmãos Paternos':
21, 'Tio Avo Materno': 162, 'Sobrinho Neto Materno': 145, 'Avo Materno': 39, 'Filhos': 27, 'Avo Paterno': 10, 'Neto Paterno': 8, 'Tios Paternos': 12
, 'Tio Bisavo Materno': 3, 'Primos': 13, 'Parente': 4, 'Primos Paternos': 1, 'Irmãos Maternos': 4, 'Sobrinhos Netos Paternos': 2, 'Sobrinho Neto': 2,
'Tio Avo': 2, 'Tio Bisavo Paterno': 1, 'Sobrinhos Netos Maternos': 5, 'Sobrinho Bisneto Materno': 3, 'Primos Maternos': 1}
Qual o parentesco: Pai
Parentesco: Pai Resultado: 468
Qual o parentesco: Filho
Parentesco: Filho Resultado: 345
Qual o parentesco: Primos Maternos
Parentesco: Primos Maternos Resultado: 1
Qual o parentesco: Tio Avo Materno
Parentesco: Tio Avo Materno Resultado: 162
Qual o parentesco: Sobrinho
Não existe parentesco
Qual o parentesco: Irmão
Parentesco: Irmão Resultado: 12350
Qual o parentesco: []
```

2.2.4 Alínea d)

```
Pratico > Ex1 > {} sample.json > ...
{
  "0": {
    "numero": "575",
    "ano": "1894-11-08",
    "pessoas": [
      {
        "nome": "Aarao Pereira Silva"
      },
      {
        "nome": "Antonio Pereira Silva"
      },
      {
        "nome": "Francisca Campos Silva"
      }
    ]
  },
  "1": {
    "numero": "582",
    "ano": "1909-05-12",
    "pessoas": [
      {
        "nome": "Abel Almeida"
      },
      {
        "nome": "Antonio Manuel Almeida"
      },
      {
        "nome": "Teresa Maria Sousa"
      }
    ]
  },
  "2": {
    "numero": "583"
  }
}
```


Capítulo 3

Exercício 5

3.1 Análise e concepção da resolução do problema

Neste exercício é nos proposto fazer um conversor de um ficheiro CSV para o formato JSON. Usamos na concepção do problema os módulos re e json.

Quando se trata de um arquivo CSV a primeira coisa que é necessario fazer é separar a primeira linha do arquivo e aplicamos a função search do modulo re a esta linha com a expressão regular:

```
r'::([A-Za-z]+)'
```

para descobrir se temos alguma função a calcular e qual seria , depois disso aplicamos a função sub do módulo re com a seguinte expressão:

```
r'(Notas){[0-9](\,[0-9])?}(::[A-Za-z]+)?'
```

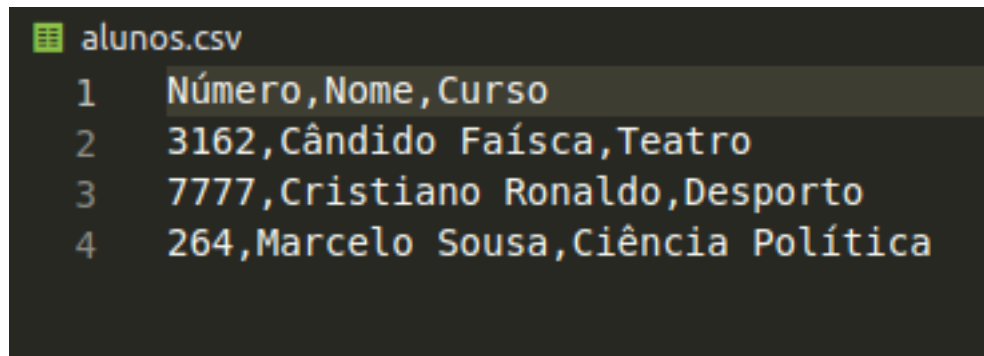
Por último fazemos o split pelo separador do documento, nesse caso ","e colocamos essa informação na variavel atributos.

Após isso percorremos linha a linha o arquivo fazendo o split pelo separador do documento o ",", fazendo corresponder a cada elemento da linha o atributo correspondete, caso haja mais q tres atributos, sabemos que um deles será as notas e damos um tratamento especial , por fim testamos se existe função e qual tipo que ela é fazemos os calculos . e adicionamos este dic a lista .

Quando termina este ciclo transformamos a lista que criamos em um json.

3.2 Testes

3.2.1 alunos.csv

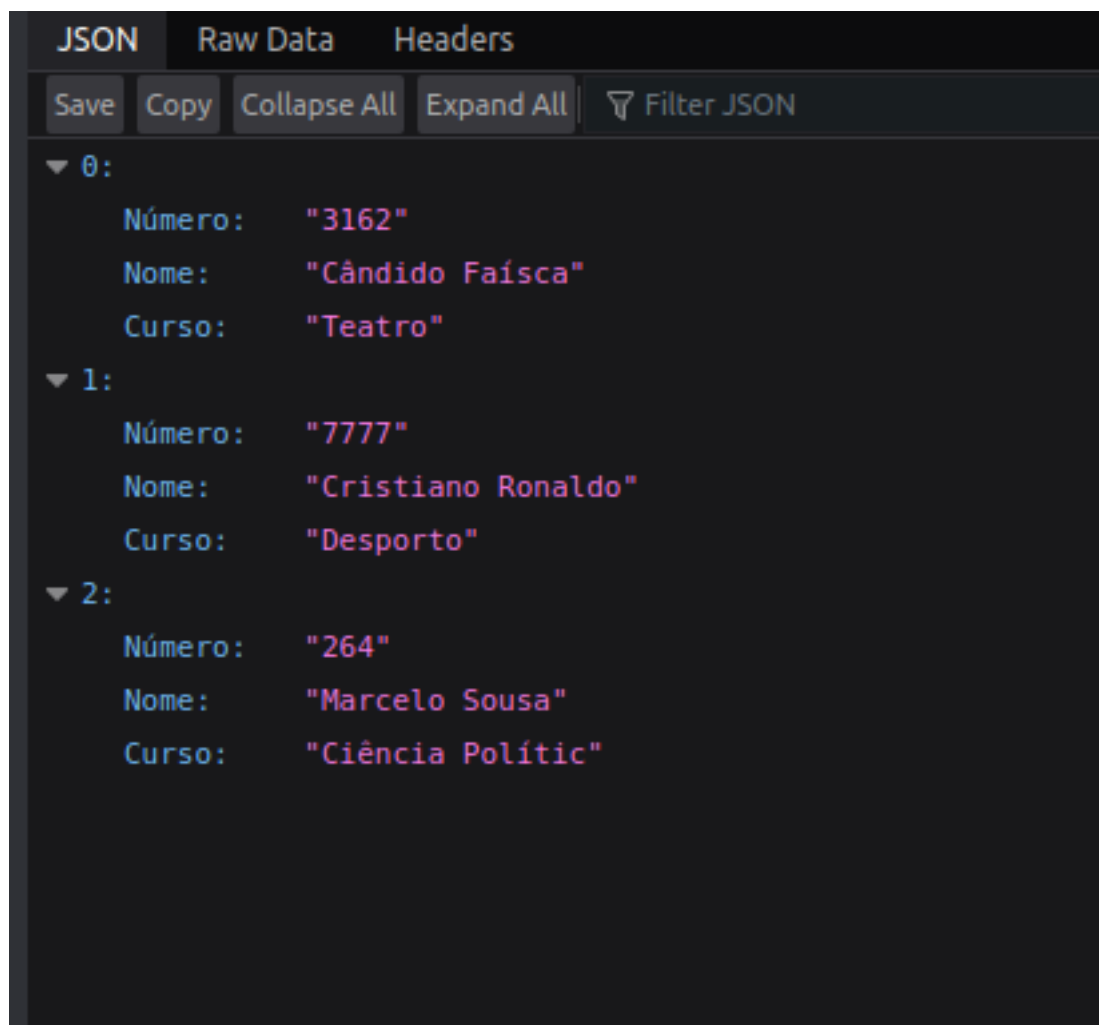


The screenshot shows a terminal window with a dark background. At the top, there is a green icon of a notepad and the filename 'alunos.csv'. Below it, the contents of the file are displayed as a CSV with four rows. The first row contains the headers 'Número', 'Nome', and 'Curso'. The subsequent three rows contain student data: '3162,Cândido Faísca,Teatro', '7777,Cristiano Ronaldo,Desporto', and '264,Marcelo Sousa,Ciência Política'.

1	Número, Nome, Curso
2	3162,Cândido Faísca,Teatro
3	7777,Cristiano Ronaldo,Desporto
4	264,Marcelo Sousa,Ciência Política

```
(base) tales@tales-MS-7C37:~/Desktop/PLC/TrabalhoPLC/Ex5$ /bin/python /home/tales/Desktop/PLC/TrabalhoPLC/Ex5/csv_to_json.py
Número, Nome, Curso
[{"Número": "Número", "Nome": "Nome", "Curso": "Curso"}]
atr: [{"Número": "Número", "Nome": "Nome", "Curso": "Curso"}]
segundo dicionário: {'Número': '3162', 'Nome': 'Cândido Faísca', 'Curso': 'Teatro'}
segundo dicionário: {'Número': '7777', 'Nome': 'Cristiano Ronaldo', 'Curso': 'Desporto'}
segundo dicionário: {'Número': '264', 'Nome': 'Marcelo Sousa', 'Curso': 'Ciência Polític'}


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



The screenshot shows a JSON viewer application with a dark theme. At the top, there are three tabs: 'JSON' (selected), 'Raw Data', and 'Headers'. Below the tabs are several buttons: 'Save', 'Copy', 'Collapse All', 'Expand All', and a 'Filter JSON' button with a magnifying glass icon. The main area displays a JSON array with three objects, indexed from 0 to 2. Each object has three key-value pairs: 'Número', 'Nome', and 'Curso'. The values are quoted strings.

```
0:
  Número: "3162"
  Nome: "Cândido Faísca"
  Curso: "Teatro"
1:
  Número: "7777"
  Nome: "Cristiano Ronaldo"
  Curso: "Desporto"
2:
  Número: "264"
  Nome: "Marcelo Sousa"
  Curso: "Ciência Polític"
```

3.2.2 alunos2.csv

 alunos2.csv

```
1  Número, Nome, Curso, Notas{5}, , , , ,
2  3162, Cândido Faísca, Teatro, 12, 13, 14, 15, 16
3  7777, Cristiano Ronaldo, Desporto, 17, 12, 20, 11, 12
4  264, Marcelo Sousa, Ciência Política, 18, 19, 19, 20, 18
5
```

```
(base) tales@tales-MS-7C37:~/Desktop/PLC/TrabalhoPLC/Ex5$ /bin/python /home/tales/Desktop/PLC/TrabalhoPLC/Ex5/csv_to_json.py
Número, Nome, Curso, Notas{5}, , , , ,
```

```
[{'Número': 'Número', 'Nome': 'Nome', 'Curso': 'Curso', 'Notas': ['', '', '', '', '']}
atr: [{'Número': 'Número', 'Nome': 'Nome', 'Curso': 'Curso', 'Notas': ''}]
{'Número': '3162', 'Nome': 'Cândido Faísca', 'Curso': 'Teatro', 'Notas': [12, 13, 14, 15]}
segundo dicionário: {'Número': '3162', 'Nome': 'Cândido Faísca', 'Curso': 'Teatro', 'Notas': [12, 13, 14, 15, 16]}
{'Número': '7777', 'Nome': 'Cristiano Ronaldo', 'Curso': 'Desporto', 'Notas': [17, 12, 20, 11]}
segundo dicionário: {'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]}
segundo dicionário: {'Número': '264', 'Nome': 'Marcelo Sousa', 'Curso': 'Ciência Política', 'Notas': [18, 19, 19, 20, 18]}
```

```
[{'Número': '3162', '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]}]
```

JSONRaw DataHeaders

SaveCopyCollapse AllExpand AllFilter

▼ 0:

Número: "3162"

Nome: "Cândido Faísca"

Curso: "Teatro"

▼ Notas:

0: 12

1: 13

2: 14

3: 15

4: 16

▼ 1:

Número: "7777"

Nome: "Cristiano Ronaldo"

Curso: "Desporto"

▼ Notas:

0: 17

1: 12

2: 20

3: 11

4: 12

▼ 2:

Número: "264"

Nome: "Marcelo Sousa"

Curso: "Ciência Política"

▼ Notas:

0: 18

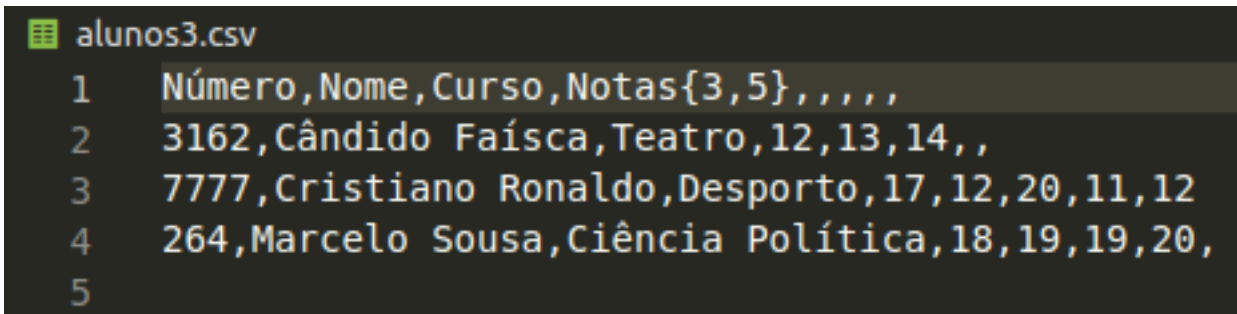
1: 19

2: 19

3: 20

4: 18

3.2.3 alunos3.csv



```
alunos3.csv
1  Número, Nome, Curso, Notas{3,5}, , , , ,
2  3162, Cândido Faísca, Teatro, 12, 13, 14, ,
3  7777, Cristiano Ronaldo, Desporto, 17, 12, 20, 11, 12
4  264, Marcelo Sousa, Ciência Política, 18, 19, 19, 20,
5
```

```
(base) tales@tales-WS-7C37:~/Desktop/PLC/TrabalhoPLC/Ex5$ /bin/python /home/tales/Desktop/PLC/TrabalhoPLC/Ex5/csv_to_json.py
Número, Nome, Curso, Notas{3,5}, , , , ,
Número, Nome, Curso, Notas{3,5}, , , , ,
[{'Número': 'Número', 'Nome': 'Nome', 'Curso': 'Curso', 'Notas': ['', '', '', '', '']}
atr: {'Número': 'Número', 'Nome': 'Nome', 'Curso': 'Curso', 'Notas': ''}
segundo dicionário: {'Número': '3162', 'Nome': 'Cândido Faísca', 'Curso': 'Teatro', 'Notas': [12, 13, 14]}
{'Número': '7777', 'Nome': 'Cristiano Ronaldo', 'Curso': 'Desporto', 'Notas': [17, 12, 20, 11]}
segundo dicionário: {'Número': '7777', 'Nome': 'Cristiano Ronaldo', 'Curso': 'Desporto', 'Notas': [17, 12, 20, 11, 12]}
segundo dicionário: {'Número': '264', 'Nome': 'Marcelo Sousa', 'Curso': 'Ciência Política', 'Notas': [18, 19, 19, 20]}

[{'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]}]
```

JSONRaw DataHeaders

SaveCopyCollapse AllExpand AllFilter JSON

▼ 0:

Número:

"3162"

Nome:

"Cândido Faísca"

Curso:

"Teatro"

▼ Notas:

0:

12

1:

13

2:

14

▼ 1:

Número:

"7777"

Nome:

"Cristiano Ronaldo"

Curso:

"Desporto"

▼ Notas:

0:

17

1:

12

2:

20

3:

11

4:

12

▼ 2:

Número:

"264"

Nome:

"Marcelo Sousa"

Curso:

"Ciência Política"

▼ Notas:

0:

18

1:

19


2:

19

3:

20

3.2.4 alunos4.csv

 alunos4.csv

```
1  Número, Nome, Curso, Notas{3,5}::sum,,,,,
2  3162, Cândido Faísca, Teatro, 12, 13, 14,,
3  7777, Cristiano Ronaldo, Desporto, 17, 12, 20, 11, 12
4  264, Marcelo Sousa, Ciência Política, 18, 19, 19, 20,
5
```

```
Número, Nome, Curso, Notas{3,5}::sum,,,,,
```

```
[['Número', 'Nome', 'Curso', 'Notas', '', '', '', '', '']
```

```
atr: [['Número', 'Nome', 'Curso', 'Notas', 'sum']
```

```
segundo dicionário: {'Número': '3162', 'Nome': 'Cândido Faísca', 'Curso': 'Teatro', 'Notas': [12, 13, 14], 'sum': 39}
```

```
{'Número': '7777', 'Nome': 'Cristiano Ronaldo', 'Curso': 'Desporto', 'Notas': [17, 12, 20, 11]}
```

```
segundo dicionário: {'Número': '7777', 'Nome': 'Cristiano Ronaldo', 'Curso': 'Desporto', 'Notas': [17, 12, 20, 11, 12], 'sum': 72}
```

```
segundo dicionário: {'Número': '264', 'Nome': 'Marcelo Sousa', 'Curso': 'Ciência Política', 'Notas': [18, 19, 19, 20], 'sum': 76}
```

```
[{'Número': '3162', 'Nome': 'Cândido Faísca', 'Curso': 'Teatro', 'Notas': [12, 13, 14], 'sum': 39}, {'Número': '7777', 'Nome': 'Cristiano Ronaldo', 'Curso': 'Desporto', 'Notas': [17, 12, 20, 11, 12], 'sum': 72}, {'Número': '264', 'Nome': 'Marcelo Sousa', 'Curso': 'Ciência Política', 'Notas': [18, 19, 19, 20], 'sum': 76}]
```

JSONRaw DataHeaders

SaveCopyCollapse AllExpand All

Filter

▼ 0:

Número:"3162"

Nome:"Cândido Faísca"

Curso:"Teatro"

▼ Notas:

0:12

1:13

2:14

sum:39

▼ 1:

Número:"7777"

Nome:"Cristiano Ronaldo"

Curso:"Desporto"

▼ Notas:

0:17

1:12

2:20

3:11

4:12

sum:72

▼ 2:

Número:"264"

Nome:"Marcelo Sousa"

Curso:"Ciência Política"

▼ Notas:

0:18

1:19

2:19

3:20

sum:76

3.2.5 alunos5.csv

```
alunos5.csv
1  Número, Nome, Curso, Notas{3,5}::media,,,,
2  3162, Cândido Faísca, Teatro, 12, 13, 14,,
3  7777, Cristiano Ronaldo, Desporto, 17, 12, 20, 11, 12
4  264, Marcelo Sousa, Ciência Política, 18, 19, 19, 20,
```

```
(base) tales@tales-MS-7C37:~/Desktop/PLC/TrabalhoPLC/Ex5$ /bin/python /home/tales/Desktop/PLC/TrabalhoPLC/Ex5/csv_to_json.py
Número, Nome, Curso, Notas{3,5}::media,,,,
```

```
[{'Número': 'Número', 'Nome': 'Nome', 'Curso': 'Curso', 'Notas': ['', '', '', '', ''], 'media': ''}]
atr: [{'Número': 'Número', 'Nome': 'Nome', 'Curso': 'Curso', 'Notas': 'media'}]
segundo dicionário: {'Número': '3162', 'Nome': 'Cândido Faísca', 'Curso': 'Teatro', 'Notas': [12, 13, 14], 'media': 13.0}
{'Número': '7777', 'Nome': 'Cristiano Ronaldo', 'Curso': 'Desporto', 'Notas': [17, 12, 20, 11]}
segundo dicionário: {'Número': '7777', 'Nome': 'Cristiano Ronaldo', 'Curso': 'Desporto', 'Notas': [17, 12, 20, 11, 12], 'media': 14.4}
segundo dicionário: {'Número': '264', 'Nome': 'Marcelo Sousa', 'Curso': 'Ciência Política', 'Notas': [18, 19, 19, 20], 'media': 19.0}
```

```
[{'Número': '3162', 'Nome': 'Cândido Faísca', 'Curso': 'Teatro', 'Notas': [12, 13, 14], 'media': 13.0}, {'Número': '7777', 'Nome': 'Cristiano Ronaldo', 'Curso': 'Desporto', 'Notas': [17, 12, 20, 11, 12], 'media': 14.4}, {'Número': '264', 'Nome': 'Marcelo Sousa', 'Curso': 'Ciência Política', 'Notas': [18, 19, 19, 20], 'media': 19.0}]
```

JSON	Raw Data	Headers
Save	Copy	Collapse All
	Expand All	Filter
▼ 0:		
Número:	"3162"	
Nome:	"Cândido Faísca"	
Curso:	"Teatro"	
▼ Notas:		
0:	12	
1:	13	
2:	14	
media:	13	
▼ 1:		
Número:	"7777"	
Nome:	"Cristiano Ronaldo"	
Curso:	"Desporto"	
▼ Notas:		
0:	17	
1:	12	
2:	20	
3:	11	
4:	12	
media:	14.4	
▼ 2:		
Número:	"264"	
Nome:	"Marcelo Sousa"	
Curso:	"Ciência Política"	
▼ Notas:		
0:	18	
1:	19	
2:	19	
3:	20	
media:	19	

Capítulo 4

Conclusão

Este projeto foi do mais importante para a introdução a técnicas na área do processamento de linguagens. Permitiu-nos aplicar em casos concretos os nossos conhecimentos de expressões regulares pelo módulo `re` do python com o uso das funções `'split'`, `'search'`, `'match'`, `'sub'` e `'findall'`, através do manuseamento de vários tipos de ficheiros diferentes e a sua conversão, tratamento de anomalias nos arquivos de maneira desejada e a manipulação dos dados dentro dos ficheiros.

Apêndice A

Código do Programa

Lista-se a seguir o código dos dois exercícios resolvidos.

A.1 Exercício 1

A.1.1 Alínea a)

```
import re

f = open("processos.txt", "r")
ant = ""
dic = {}
ano = ""
for linha in f:
    if ant != linha:
        y = re.match(r'([1-9][0-9]*):([0-9]{4})', linha)
        if y:
            ano = y.group(2)
            if ano not in dic:
                dic[ano] = 1
            else:
                dic[ano] += 1
    ant = linha

frase = input("Qual o ano? ")
while frase != "":
    if frase in dic:
        print("Ano: ", frase, "Resultado: ", dic[frase])
    else:
        print("Não existe processo")
    frase = input("Qual o ano?")
print(dic)
```

A.1.2 Alínea b)

```
import re

f = open("processos.txt", "r")
ant = ""
dic_nome = {}
dic_apelido = {}
ano = ""
info = []
nome_proprio = ""
nome_apelido = ""
seculo = 0

def add_dic(pessoa):

    if "Doc.danificado" not in pessoa:
        nome = re.match(rf'([A-Z][a-z]+)([ ]+[A-Z][a-z]+)',pessoa)
        if nome:
            nome_proprio = nome.group(1)
            nome_apelido = nome.group(2).lstrip()

            if nome_proprio not in dic_nome[seculo]:
                dic_nome[seculo][nome_proprio] = 1
            else:
                dic_nome[seculo][nome_proprio] += 1

            if nome_apelido not in dic_apelido[seculo]:
                dic_apelido[seculo][nome_apelido] = 1
            else:
                dic_apelido[seculo][nome_apelido] += 1

for linha in f:
    if ant != linha:
        info = re.split(r':+',linha)
        #print(info)
        if info[0] != "\n":
            ano = info[1][0:4]
            #print(ano)
            if ano[2:4] == "00":
                seculo = int(ano[0:2])
            else:
                seculo = int(ano[0:2]) + 1

            if seculo not in dic_nome:
                dic_nome[seculo] = {}
            if seculo not in dic_apelido:
                dic_apelido[seculo] = {}

            for elem in info[2:-1]:
                parent_proc = re.findall(r'([a-zA-Z ]+)(\,[a-zA-Z ]+)(\.[ ]Proc\.[0-9]+)',elem)
                if parent_proc:
                    for pessoa in parent_proc:
                        add_dic(pessoa[0])
                else:
                    add_dic(elem)

frase = input("Qual o seculo? ")
while frase != "":
```

```

seculo = int(frase)
nome = input("Qual o nome? ")
apelido = input("Qual o apelido? ")
if seculo in dic_nome and int(frase) in dic_apelido:
    #print("Seculo certo!!!")
    if nome in dic_nome[seculo] and apelido in dic_apelido[seculo]:
        print("Seculo: ",seculo, "Nome: ",dic_nome[seculo][nome], "Apelido:
        ↪ ",dic_apelido[seculo][apelido])
    else:
        print("Não existe nome ou apelido neste século")
else:
    print("Não existe processo")
frase = input("Qual o seculo? ")

```

A.1.3 Alínea c)

```

import re

f = open("processos.txt", "r")

parentesco = {}

aux = ''
for linha in f:
    info = re.split(r'::+',linha)
    if len(info)> 4:
        aux = re.findall(r'\,((([A-Z][a-z]+[ ]*)+)\. [ ]*Proc',info[-2])
        for pare in aux:
            if pare[0] not in parentesco:
                parentesco[pare[0]] = 1
            else:
                parentesco[pare[0]] += 1

print(parentesco)
frase = input("Qual o parentesco: ")
while frase != "":
    if frase in parentesco:
        print("Parentesco: ",frase, "Resultado: ",parentesco[frase])
    else:
        print("Não existe parentesco")
    frase = input("Qual o parentesco: ")

```

A.1.4 Alínea d)

```
import re, json

f = open("processos.txt", "r")
dic_linhas = {}
linha = ""
parent_proc = []
dic_pessoa = {}

for i in range(20):
    dic_linhas[i] = {}
    linha = next(f)
    info = re.split(r':+', linha)
    dic_linhas[i]["numero"] = info[0]
    dic_linhas[i]["ano"] = info[1]
    dic_linhas[i]["pessoas"] = []
    for elem in info[2:-1]:
        #print(elem)
        parent_proc = re.findall(r'([a-zA-Z ]+)(\,[a-zA-Z ]+)(\.[ ]Proc\.[0-9]+)', elem)
        #print(parent_proc)
        if parent_proc:
            for pessoa in parent_proc:
                dic_pessoa = {"nome" : pessoa[0], "parentesco" : pessoa[1][1:], "processo" : pessoa[2][7:]}
                dic_linhas[i]["pessoas"].append(dic_pessoa)
        else:
            dic_pessoa = {"nome" : elem}
            dic_linhas[i]["pessoas"].append(dic_pessoa)

print(dic_linhas)
json_object = json.dumps(dic_linhas, indent=4)

with open("sample.json", "w") as outfile:
    outfile.write(json_object)
```

A.2 Exercício 2

```
import re, json

f = open("alunos5.csv", "r")
list = []
primeira = next(f)
print(primeira)
funcao = re.search(r'::([A-Za-z]+)', primeira)
primeira = re.sub(r'(Notas){[0-9](\, [0-9])?}(:[A-Za-z]+)?', r'\1', primeira)

atributos = re.split(',', primeira)
atributos[-1] = atributos[-1][: -1]
print(atributos)
a = 0
for atributo in atributos:
    if atributo == '':
        break
    a += 1

if funcao:
    atributos[a] = funcao.group(1)
print("atr: ", atributos[:a+1])

for linha in f:
    dic = {}
    #print("linha: ", linha)
    valores = re.split(',', linha)
    #print("valores: ", valores)
    for j in range(a-1):
        dic[atributos[j]] = valores[j]
    if len(atributos) > 3:
        dic[atributos[a-1]] = []
        #print("dicionario: ", dic)
        for j in range(a-1, len(valores)-1):
            if valores[j] != '':
                dic[atributos[a-1]].append(int(valores[j]))
        if valores[-1] != "\n" and valores[-1] != '':
            print(dic)
            dic[atributos[a-1]].append(int(valores[-1][0:-1]))

        if atributos[a] == 'sum':
            dic[atributos[a]] = sum(dic[atributos[a-1]])
        elif atributos[a] == 'media':
            dic[atributos[a]] = sum(dic[atributos[a-1]])/len(dic[atributos[a-1]])
    else:
        #print(dic)
        dic[atributos[a-1]] = valores[a-1][: -1]
    list.append(dic)
    print("segundo dicionario: ", dic)

print("\n\n", list)

json_object = json.dumps(list, indent=4)

with open("Notas.json", "w") as outfile:
    outfile.write(json_object)
```