



**Universidade do Minho**

UNIVERSIDADE DO MINHO

LICENCIATURA EM CIÊNCIAS DA COMPUTAÇÃO

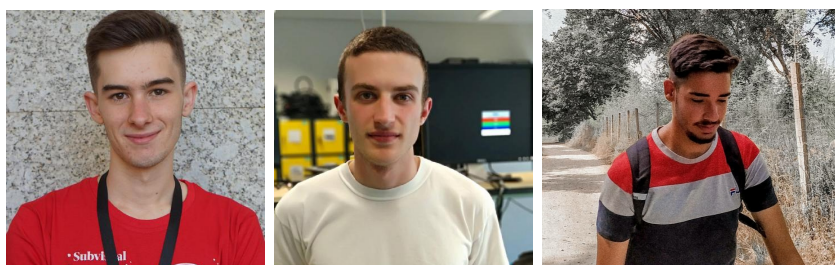
PLC - Trabalho Prático 1  
Grupo nº5

Simão Pedro Batista Caridade Quintela  
(A97444)

David José de Sousa Machado  
(A91665)

Hugo Filipe de Sá Rocha  
(A96463)

13 de novembro de 2022



# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>3</b>
<b>2</b>	<b>Enunciados e Resoluções</b>	<b>4</b>
2.1	Processador de Pessoas listadas nos Róis de Confessados . . .	4
2.1.1	Resolução do problema . . . . .	4
<b>3</b>	<b>Exemplos de funcionamento</b>	<b>11</b>
3.1	Processador de Pessoas listadas nos Róis de Confessados . . .	11
3.1.1	Frequência de processos por ano . . . . .	11
3.1.2	Frequência de nomes por século . . . . .	12
3.1.3	Frequência de relações por século . . . . .	12
3.1.4	Escrever as 20 primeiras linhas num JSON . . . . .	13

# Capítulo 1

## Introdução

No âmbito da disciplina de Processamento de Linguagens e Compiladores foi-nos proposto pelo docente Pedro Rangel Henriques a realização de um trabalho prático com o objetivo de colocar em prática a utilização de expressões regulares para a análise de ficheiros de texto.

O trabalho prático consiste na resolução de, pelo menos, um problema em cinco propostos. Analisados os problemas acabamos por resolver o problema 1 (Processador de Pessoas listadas nos Róis de Confessados) e o problema 5 (Ficheiros CSV com listas e funções de agregação).

Neste documento estão apresentadas as soluções utilizadas para a resolução dos problemas abordados bem como a correspondente demonstração do funcionamento dos programas.

## Capítulo 2

# Enunciados e Resoluções

### 2.1 Processador de Pessoas listadas nos Róis de Confessados

Construa agora um ou vários programas Python para processar o texto 'processos.txt' com o intuito de calcular frequências de alguns elementos (a ideia é utilizar arrays associativos para o efeito) conforme solicitado a seguir.

- a) Calcula a frequência de processos por ano (primeiro elemento da data);
- b) Calcula a frequência de nomes próprios (o primeiro em cada nome) e apelidos (o ultimo em cada nome) por séculos;
- c) Calcula a frequência dos vários tipos de relação: irmão, sobrinho, etc.
- d) Imprimir os 20 primeiros registos num novo ficheiro de output mas em formato JSON.

#### 2.1.1 Resolução do problema

Neste trabalho decidimos utilizar uma estratégia inicial semelhante em todas as alíneas que consiste em ler todas as linhas do ficheiro *processos.txt* e guarda-las num array para depois poderem ser lidas na resolução das diferentes alíneas.

- a. Na resolução da alínea a. utilizamos um dicionário *years* para guardar a frequência de processos por ano associando cada ano à frequência de processos. Para obtermos o ano e processo correspondente a cada linha do ficheiro utilizamos as funções **search** e **split**, e a seguinte expressão regular `([0-9]+)[:2]([0-9]4[0-9]2[0-9]2)`

Listing 2.1: Alínea a

```

1 def processFrequency():
2     years = {}
3
4     for line in lines:
5         process_and_date = re.search('([0-9]+)
6         [:]{2}([0-9]{4}\-[0-9]{2}\-[0-9]{2})', line)
7
8         if process_and_date != None:
9             process = process_and_date.group(1)
10            date = process_and_date.group(2)
11            data_splitted = re.split(r'-', date)
12            year = data_splitted[0]
13
14            if year not in years:
15                years[year] = 1
16            else:
17                years[year] += 1
18
19    return years
20
21

```

- b. Na resolução da alínea b. utilizamos o dicionário *centurys* que tem a seguinte estrutura:

```

1 # this is just an example:
2 centurys = {
3     19: {
4         "First": {"Candido": 10},
5         "Last": {"Faisca": 5}
6     },
7     20: {
8         "First": {"Ivone": 130},
9         "Last": {"Costa": 2000}
10    }
11 }
12 }
13

```

Utilizamos a expressão regular `([0-9]4)([0-9]2)([0-9]2)` com a função **search** para saber qual a data da linha lida.

Com a expressão regular `:[A-Za-z ]+(:)` e com a função **findall** conseguimos identificar o nome da pessoa processada, do pai e da mãe.

Por fim, utilizando de novo a função **findall** e a expressão regular `([A-Z] [A-Za-z ]+),([A-Za-z ]+). ?(?i:(Proc.[0-9]+))` conseguimos identificar os nomes de pessoas que tiveram envolvidas noutros processos.

Listing 2.2: Alínea b

```

1 def year_to_century(year):
2     return -(-year // 100)
3
4 def nameFrequency():
5     centurys = {}
6     for line in lines:
7         date = re.search(r'([0-9]{4})\-([0-9]{2})
8         \-([0-9]{2})', line)
9         names_in_dots = re.findall('([A-Za-z ]+)(:)',
10         line)
11         names_with_procs = re.findall('([A-Z][A-Za-z ]+
12         ,([A-Za-z ]+). ?(?i:(Proc.[0-9]+))', line)
13
14         names = names_in_dots + names_with_procs
15
16         if date:
17             year = int(date.group(1))
18             century = year_to_century(year)
19             if century not in centurys:
20                 centurys[century] = {}
21                 centurys[century]["First"] = {}
22                 centurys[century]["Last"] = {}
23
24         for name in names:
25             person_name = name[0]
26             name_splitted = re.split(" ", person_name)
27             first_name = name_splitted[0]
28             last_name = name_splitted[-1]
29             if first_name not in centurys[century]["First"]
30             ]:
31                 centurys[century]["First"][first_name] = 1
32             else:
33                 centurys[century]["First"][first_name] +=
34                 1
35
36             if last_name not in centurys[century]["Last"]:
37                 centurys[century]["Last"][last_name] = 1
38             else:
39                 centurys[century]["Last"][last_name] += 1
40         return centurys

```

- c. Na resolução da alínea c utilizamos um dicionário *rel\_freq* para guardar a frequência de relações existente no ficheiro de texto.

Ao analisar o ficheiro reparamos que estes dois padrões que se repetiam:

(a) ::Filho::Pai(opcional)::Mãe(Opcioal)::

(b) nome, relação parentesco, Proc.x

Para identificar o padrão (a) usamos a função **findall** e a expressão regular `:[A-Za-z| ]+):` . De notar que, no dicionário, o **Pai** e a **Mãe** são ambos contabilizados na entrada **Progenitor** visto que em várias linhas, por vezes a ordem pela qual aparece o nome dos mesmos é trocada. Para o efeito, e para não correr o risco de recolher informação errada optamos por coloca-los na mesma entrada.

Para identificar o padrão (b), utilizamos a função **findall** e a expressão regular `([A-Z][A-Za-z ]+),([A-Za-z ]+).?(?i:(Proc.[0-9]+))` para identificar as restantes relações de parentesco com o processado.

Listing 2.3: Alínea c

```
1 def relationFrequency():
2     rel_freq = {}
3     rel_freq["Progenitores"] = 0
4     rel_freq["Filho"] = 0
5
6     for line in lines:
7         parents_and_son = re.findall(":[A-Za-z| ]+):",
8             line)
9
10        if parents_and_son:
11            parents = parents_and_son[1:]
12            rel_freq["Filho"] += 1
13            rel_freq["Progenitores"] += len(parents)
14
15        relations = re.findall("([A-Z][A-Za-z ]+),([A-Za-z
16        ]+).?(?i:(Proc.[0-9]+))", line)
17        if relations:
18            for relation in relations:
19                if relation[1] not in rel_freq:
20                    rel_freq[relation[1]] = 1
21                else:
22                    rel_freq[relation[1]] += 1
23
24    return rel_freq
```

- d. Para fechar o exercício 1 falta imprimir as primeiras 20 linhas do ficheiro *processos.txt* em formato JSON.

Para a resolução desta alínea utilizamos duas funções, uma para recolher informação, e outra para escrever informação no ficheiro JSON pretendido.

A função **info\_to\_json** tem como objetivo recolher informação linha a linha (assegurando-se que não está a ler duas vezes a mesma linha),

utilizando expressões regulares mostradas anteriormente, na seguinte forma:

Listing 2.4: Estrutura de dados alínea d

```
1 # dada a seguinte linha tem-se
2 # 569::1867-05-23::Abel Alves Barroso::Antonio Alves
   Barroso::Maria Jose Alvares Barroso::Bento Alvares
   Barroso,Tio Paterno. Proc.32057. Domingos Jose
   Alvares Barroso,Tio Materno. Proc.32235.:
3
4
5 json_info = {
6     575::1894-11-08 :{
7         "Processo": "575",
8         "Data": "1894-11-08",
9         "Pessoa processada": "Abel Alves Barroso",
10        "Pai": "Antonio Alves Barroso",
11        "Mae": "Maria Jose Alvares Barroso",
12        "Tio Paterno": "Bento Alvares Barroso",
13        "Tio Materno": "Domingos Jose Alvares Barroso"
14    }
15
```

Listing 2.5: Função info\_to\_json

```
1 def info_to_json():
2     json_info = {}
3
4     valid_lines = 0
5     i = 0
6     while valid_lines < 20:
7         line = lines[i]
8
9         if line != '':
10            process_and_date = re.search('([0-9]+)
11            [:]{2}([0-9]{4}\-[0-9]{2}\-[0-9]{2})', line)
12            both = process_and_date.group(0)
13
14            if both not in json_info:
15                process = process_and_date.group(1)
16                date = process_and_date.group(2)
17
18                json_info[both] = {"Processo": process, "
19                Data": date}
20
21                son_and_parents = re.findall('([A-Za-z|
22                ]+)(:)', line)
23
24                json_info[both]["Pessoa processada"] =
25                son_and_parents[0][0]
26
27                if len(son_and_parents) == 2:
```



```

25         json_info[both]["Mae"] =
son_and_parents[1][0]
26     else:
27         json_info[both]["Pai"] =
son_and_parents[1][0]
28         json_info[both]["Mae"] =
son_and_parents[2][0]
29
30         relations = re.findall('([A-Z][A-Za-z ]+),([A-Za-z ]+). ?(?i:(Proc.[0-9]+))', line)
31         if relations:
32             for relation in relations:
33                 json_info[both][relation[1]] =
relation[0]
34
35
36         valid_lines+=1
37
38     i+=1
39     return json_info
40

```

Posto isto, e tendo toda a informação necessária, basta utilizar a função definida por nós **write\_on\_json** para escrever toda a informação num ficheiro JSON.

Listing 2.6: Função info\_to\_json

```

1 def write_on_json():
2     json_info = info_to_json()
3     f = open('res.json', 'w')
4
5     f.write('[\n')
6
7
8     for (i,entry) in enumerate(json_info):
9         f.write('    {\n')
10        data = json_info[entry]
11        for (j, key) in enumerate(data):
12            f.write(f'        "{key}": "{data[key]}"')
13
14        if j == len(data)-1:
15            f.write('\n')
16        else:
17            f.write(',\n')
18
19
20    if i == len(json_info)-1:
21        f.write('    }\n')
22    else:
23        f.write('    },\n')
24
25    f.write(']\n')

```

```
26     f.close()  
27
```

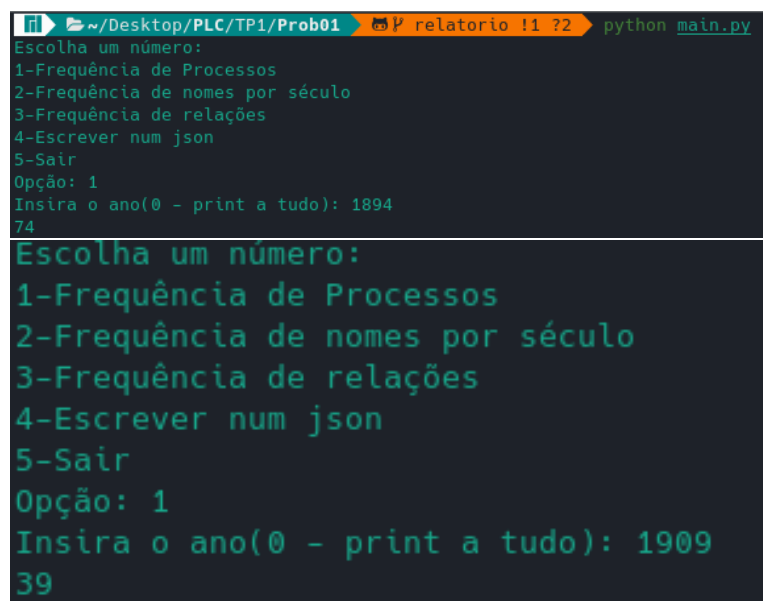
## Capítulo 3

# Exemplos de funcionamento

### 3.1 Processador de Pessoas listadas nos Róis de Confessados

Nesta secção vamos mostrar o funcionamento do programa bem como a informação recolhida pelas funções previamente apresentadas.

#### 3.1.1 Frequência de processos por ano



```
~/Desktop/PLC/TP1/Prob01 relatorio !1 ?2 python main.py
Escolha um número:
1-Frequência de Processos
2-Frequência de nomes por século
3-Frequência de relações
4-Escriver num json
5-Sair
Opção: 1
Insira o ano(0 - print a tudo): 1894
74
Escolha um número:
1-Frequência de Processos
2-Frequência de nomes por século
3-Frequência de relações
4-Escriver num json
5-Sair
Opção: 1
Insira o ano(0 - print a tudo): 1909
39
```

Como podemos ver o programa está a fazer a contagem do número de processos por ano.

### 3.1.2 Frequência de nomes por século

```

~/Desktop/PLC/TP1/Prob01  P P relatorio !! ?3  python main.py
Escolha um número:
1-Frequência de Processos
2-Frequência de nomes por século
3-Frequência de relações
4-Escriver num json
5-Sair
Opção: 2
Século(0 - print a tudo): 18
({'First': {'Acacio': 2, 'Bento': 1188, 'Josefa': 397, 'Adao': 5, 'Domingos': 3695, 'Mariana':
oa': 123, 'Francisco': 5424, 'Isabel': 1179, 'Mecia': 14, 'Angelica': 64, 'Paulo': 384, 'Dom
', 'Pedro': 1399, 'Felicia': 50, 'Angela': 308, 'Adriao': 10, 'Alexandre': 420, 'Margarida':
eotnio': 5, 'Geraldo': 71, 'Senhorinha': 230, 'Antonio': 7520, 'Catarina': 920, 'Gaspar':
tania': 22, 'Teresa': 579, 'Agostinho': 253, 'Helena': 155, 'Bras': 119, 'Luís': 1489, 'Toma
Inacio': 200, 'Antonia': 601, 'Pascoal': 156, 'Cecilia': 35, 'Rosa': 511, 'Jeronimo': 693,
16, 'Inocencia': 28, 'Luísa': 676, 'Cristina': 24, 'Tinoteo': 2, 'Marcos': 101, 'Apolonia':
', 'Last': {'Carvalho': 2200, 'Maria': 383, 'Azevedo': 946,
', 'Araujo': 2260, 'Barbosa': 1073, 'Duarte': 195, 'Jorge
Sousa': 1887, 'Bacelar': 178, 'Costa': 2437, 'Pereira':
5, 'Brandao': 282, 'Gomes': 1202, 'Afonseca': 337, 'Cape
186, 'Falcao': 92, 'Filipe': 17, 'Mendonca': 100, 'Sot
es': 282, 'Ribeiro': 1388, 'Almeida': 389, 'Coelho': 614
zerra': 25, 'Santo': 16, 'Veloso': 216, 'Murca': 2, 'Mar
', 'Leitao': 154, 'Joao': 200, 'Cruz': 308, 'Cabecas': 18

```

Como podemos ver o programa está a fazer a contagem do número de nomes e apelidos por século.

### 3.1.3 Frequência de relações por século

```
Relação(0 - print a tudo): 0
{'Progenitores': 75607, 'Filho': 38225, 'Tio Paterno': 1853, 'Tio Materno': 1853, 'Sobrinho Paterno': 1635, 'Irmaos': 686, 'Sobrinhos Maternos': 98, 'Irmão Paterno': 638, 'Primo Materno': 225, 'Tio Avo Paterno': 98, 'Irmão Materno': 98, 'Sobrinho Materno': 162, 'Sobrinho Neto Materno': 145, 'Avo Materno': 39, 'Filhos': 39, 'Primos': 13, 'Parente': 4, 'Primos Paternos': 1, 'Irmaos Maternos': 4, 'Sobrinho Paterno': 1, 'Sobrinhos Netos Maternos': 5, 'Sobrinho Bisneto Materno': 3, 'Primo Avo': 1}

Escolha um número:
1-Frequência de Processos
2-Frequência de nomes por século
3-Frequência de relações
4-Escriver num json
5-Sair
Opção: 3
Relação(0 - print a tudo): Irmaos
686
```

Como podemos ver o programa está a fazer a contagem do número de relações.

### 3.1.4 Escrever as 20 primeiras linhas num JSON

```
[
  {
    "Processo": "575",
    "Data": "1894-11-08",
    "Pessoa processada": "Aarao Pereira Silva",
    "Pai": "Antonio Pereira Silva",
    "Mãe": "Francisca Campos Silva"
  },
  {
    "Processo": "582",
    "Data": "1909-05-12",
    "Pessoa processada": "Abel Almeida",
    "Pai": "Antonio Manuel Almeida",
    "Mãe": "Teresa Maria Sousa"
  },
  {
    "Processo": "569",
    "Data": "1867-05-23",
    "Pessoa processada": "Abel Alves Barroso",
    "Pai": "Antonio Alves Barroso",
    "Mãe": "Maria Jose Alvares Barroso",
    "Tio Paterno": "Bento Alvares Barroso",
    "Tio Materno": "Domingos Jose Alvares Barroso"
  },
]
```

Como podemos ver o programa está a escrever corretamente no ficheiro JSON.