Processamento de Linguagens e Compiladores

Universidade do Minho
Ciências da Computação
Trabalho Prático 1

Processador de Pessoas Listadas em Róis de Confessados

João Pedro Carvalho Henriques
A81705

2022

Conteúdo

1 Introdução

O seguinte relatório foi realizado no âmbito da disciplia de Processamento de Linguagens e Compiladores onde são expostas todas as decisões tomadas na realização do trabalho prático.

O trabalho consiste no desenvolvimento de filtros de texto para extrair informação de um ficheiro de texto e tem como objetivo consolidar a aprendizagem da utilização de linguagens regulares em programas usando o módulo "re"do Python.

2 Problema

2.1 Descrição do problema

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.

2.2 Análise e Abordagem ao problema

Após uma breve leitura do ficheiro processos.txt é possível observar que cada linha do ficheiro contém vários campos de informação separados por '::'. Analisando várias linhas foi possível verificar a existência de um padrão. Esse padrão pode ser definido pelos seguintes campos: número, data, nome, nome do pai, nome da mãe e informação adicional.

Após esta análise, já é possivel fazer uma abordagem mais concreta na resolução das alíneas. Podemos concluir que, '::' antecede aos nomes próprios e procede aos apelidos, os nomes que aparecem no campo da informação adicional não foram contabilizados. As relações aparecem no campo da informação adicional sempre precedidas por uma vírgula.

Assim,foi criado um menu inicial que permite ao utilizador escolher qual das alíneas pretende ver.Também foi criado um ficheiro de teste, teste.txt, correspondente às primeiras 30 linhas do ficheiro processos.txt, para testar as funções definidas na resolução das alíneas.

Para facilitar a visualização da informação foram desenvolvidas duas funções em Python que através de um dicionario criam um ficheiro em HTML de uma tabela com a informação do dicionario.

Este projeto é constiuído por 2 ficheiros Python. O ficheiro func.py onde estão todas as funções necessárias para a resolução das alíneas e para a criação do ficheiro HTML e o ficheiro main.py onde está a implementação do menu inicial e onde são invocadas as funções.

3 Decisões tomadas e Exemplo de teste

Nesta secção vão ser explicadas as decisões tomadas na implementação do código e o resultado da aplicação ao ficheiro teste.txt

3.1 Frequência de processos por ano

Para a resolução da primeira alínea foi utilizado um dicionário, associando a cada ano o número de processos registados. Assim, foi iterado sobre cada linha do ficheiro, e usando a função **search()** e a expressão regular **([0-9]4)-([0-9]2)-([0-9]2)** foi possivel identificar a data e associar o primeiro grupo ao ano (ano = data.group(1)). Também foi feita a verificação para contabilizar linhas com número de processo e data iguais apenas uma vez.

```
def freq_processos_ano(ficheiro):

    f = open(ficheiro, 'r')
    l = f.readlines()

    dic = {}
    proc = []
    for line in l:
        x = re.split(r'::',line)
        data = re.search(r'([0-9]{4})-([0-9]{2})-([0-9]{2})',line)

        if(data):
            if (x[0],x[1]) not in proc:
                proc.append((x[0],x[1]))

                ano = data.group(1)

                if ano not in dic:
                    dic[ano] = 1
                else:
                    dic[ano] += 1
    f.close()
    return dic
```

Figura 1: Frequênica de processos por ano

```
{'1894': 3, '1909': 2, '1867': 1, '1896': 1, '1904': 2, '1901': 1, '1883': 2, '1900': 1, '1902': 3,
'1880': 1, '1889': 1, '1908': 2, '1869': 1, '1862': 1, '1906': 1, '1856': 1, '1875': 1, '1892': 1}
```

Figura 2: Exemplo de teste, Frequência de processos por ano

Ano	Numero de processos
1894	3
1909	2
1867	1
1896	1
1904	2
1901	1
1883	2
1900	1
1902	3
1880	1
1889	1
1908	2
1869	1
1862	1
1906	1
1856	1
1875	1
1892	1

Figura 3: Tabela de teste, Frequência de processos por ano

3.2 Frequência de nomes próprios e apelidos por século

Na resolução da segunda alínea, o problema é divido em duas subalíneas, uma que calcula a frequência de nomes próprios por século e outra a frequência de apelidos por século

3.2.1 Frequência de nomes próprios por século

Para calcular a frequência de cada nome próprio por século foi também utilizado um dicionário que atribui a cada século outro dicionário que, por sua vez, atribui ao primeiro nome de cada nome a sua frequência,no respetivo século. Foi iterado sobre cada linha do ficheiro, onde através da expressão regular **'([A-Za-z]+)([ ][A-Za-z]+)*([A-Za-z]+)?', r'\1'** e da função **sub()**, todos os nomes completos são subtituidos pelo primeiro nome, e através da função **split()** aplicada ao resultado da ação anterior foi possivel guardar todos os campos separados por **'::'**. Assim os campos referentes ao nome, nome do pai e nome da mãe contêm apenas o primeiro nome.

```
def freq_nomes_proprios_sec(ficheiro):

    f = open(ficheiro, 'r')
    l = f.readlines()

    proc = []
    dic = {}

    for line in l:
        data = re.search(r'([0-9]{4})-([0-9]{2})-([0-9]{2})',line)
        y = re.sub(r'([A-Za-z]+)([ ][A-Za-z]+)*(\,[A-Za-z]+)?', r'\1', line)
        x = re.split(r'::', str(y))

        if(data):
            ano = data.group(1)
            seculo = int(ano[:2]) + 1

            if (x[0],x[1]) not in proc:
                proc.append((x[0],x[1]))

                if seculo not in dic:
                    dic[seculo] = {}
                for nome in x[2:5]:
                    if nome != "":
                        if nome not in dic[seculo]:
                            dic[seculo][nome] = 1
                        else:
                            dic[seculo][nome] += 1
    f.close()
    return dic
```

Figura 4: Frequência de nomes próprios por século

```
{19: {'Aarao': 1, 'Antonio': 2, 'Francisca': 1, 'Abel': 3, 'Maria': 4, 'Francisco': 2, 'Antonia': 1, 'J
oao': 3, 'Cecilia': 1, 'Abelardo': 1, 'Jose': 6, 'Leopoldina': 2, 'Abilio': 7, 'Sebastiao': 1, 'Teresa'
: 2, 'Ana': 1, 'Flora': 1, 'Acacio': 2, 'Albina': 1}, 20: {'Abel': 5, 'Antonio': 3, 'Teresa': 2, 'Ana':
 1, 'Jose': 2, 'Bernardina': 1, 'Serafim': 1, 'Emilia': 1, 'Constantino': 1, 'Margarida': 1, 'Abilio':
7, 'Joaquim': 2, 'Josefa': 1, 'Perpetua': 1, 'Maria': 2, 'Manuel': 2, 'Ermelinda': 1, 'Florinda': 1}}
```

Figura 5: Exemplo de teste, Frequência de nomes próprios por século

Seculo	Primeiro Nome	Frequencia
19	Aarao	1
19	Antonio	2
19	Francisca	1
19	Abel	3
19	Maria	4
19	Francisco	2
19	Antonia	1
19	Joao	3
19	Cecilia	1
19	Abelardo	1
19	Jose	6
19	Leopoldina	2
19	Abilio	7
19	Sebastiao	1
19	Teresa	2
19	Ana	1
19	Flora	1
19	Acacio	2
19	Albina	1
20	Abel	5
20	Antonio	3
20	Teresa	2
20	Ana	1
20	Jose	2
20	Bernardina	1
20	Serafim	1
20	Emilia	1
20	Constantino	1
20	Margarida	1
20	Abilio	7
20	Joaquim	2

Figura 6: Parte da Tabela de teste, Frequência de nomes próprios por século

3.2.2 Frequência de apelidos por século

A abordagem a esta alínea é bastante similar à anterior.Aqui foi utilizada a expressão regular **r'([A-Za-z]+[ ])*([A-Za-z]+)(\,[A-Za-z]+)*', r'\2'** que nos permite substituir o nome completo pelo apelido\último nome. A última parte da expressão regular, **'(\,[A-Za-z]+)*'** é necessária devido a existirem apelidos seguidos de uma virgula e estado civil, "apelido,solteira".

```
def freq_apelidos_sec(ficheiro):

    f = open(ficheiro, 'r')
    l = f.readlines()

    proc = []
    dic = {}

    for line in l:
        data = re.search(r'([0-9]{4})-([0-9]{2})-([0-9]{2})',line)
        y = re.sub(r'([A-Za-z]+[ ])*([A-Za-z]+)(\,[A-Za-z]+)*', r'\2',line)
        x = re.split(r'::',str(y))

        if(data):
            ano = data.group(1)
            seculo = int(ano[:2]) + 1

            if (x[0],x[1]) not in proc:
                proc.append((x[0],x[1]))

                if seculo not in dic:
                    dic[seculo] = {}
                for nome in x[2:5]:
                    if nome != "":
                        if nome not in dic[seculo]:
                            dic[seculo][nome] = 1
                        else:
                            dic[seculo][nome] += 1

    f.close()
    return dic
```

Figura 7: Frequência de apelidos por século

```
{19: {'Silva': 3, 'Barroso': 4, 'Oliveira': 3, 'Rebelo': 1, 'Freitas': 1, 'Pereira': 2, 'Araujo': 3, 'G
uerreiro': 2, 'Monteiro': 1, 'Alves': 2, 'Rocha': 1, 'Arantes': 2, 'Sousa': 2, 'Santos': 2, 'Jesus': 1,
 'Leite': 2, 'Fernandes': 1, 'Guimaraes': 3, 'Borges': 2, 'Teixeira': 1, 'Carneiro': 1, 'Barbosa': 2},
20: {'Almeida': 2, 'Sousa': 1, 'Reis': 4, 'Abreu': 1, 'Dantas': 1, 'Pereira': 2, 'Goncalves': 1, 'Carva
lho': 2, 'Rego': 1, 'Guimaraes': 2, 'Gomes': 2, 'Galvao': 3, 'Magalhaes': 2, 'Correia': 2, 'Imperadeiro
': 2, 'Lourenco': 1, 'Araujo': 1, 'Couto': 1, 'Fonseca': 1, 'Ferreira': 2, 'Silva': 1}}
```

Figura 8: Exemplo de teste, Frequência de apelidos por século

Seculo	Apelido	Frequencia
19	Silva	3
19	Barroso	4
19	Oliveira	3
19	Rebelo	1
19	Freitas	1
19	Pereira	2
19	Araujo	3
19	Guerreiro	2
19	Monteiro	1
19	Alves	2
19	Rocha	1
19	Arantes	2
19	Sousa	2
19	Santos	2
19	Jesus	1
19	Leite	2
19	Fernandes	1
19	Guimaraes	3
19	Borges	2
19	Teixeira	1
19	Carneiro	1
19	Barbosa	2
20	Almeida	2
20	Sousa	1
20	Reis	4
20	Abreu	1
20	Dantas	1
20	Pereira	2
20	Goncalves	1
20	Carvalho	2
20	Rego	1

Figura 9: Parte da Tabela de teste, Frequência de apelidos por século

3.3 Frequência dos tipos de relação

Na resolução deste alínea foi usada a função **split()** associada ao caracter vírgula, pois as relações aparecem sempre entre uma virgula e um ponto final. Assim, foi iterado sobre o resultado anterior a partir da primeira virgula que aparece na linha e foi usada a função **match()** associada à expressão regular '**[A-Za-z]***(*Irmao|Tio|Mae|Sobrinh[ao]|Pai|[nN]et[ao]|Meio|[Aa]vo*)**[A-Za-z]*\.**' que nos permite encontrar todas as relações e devolver o último grupo do resultado que é o nome total da relação que está entre a virgula e o ponto final.

```
def freq_relacoes(ficheiro):

    f = open(ficheiro, 'r')
    l = f.readlines()

    dic={}

    for line in l:
        x = re.split(r',',line)

        for elem in x[1:]:
            y = re.match(r'[A-Za-z]*(Irmao|Tio|Mae|Sobrinh[ao]|Pai|[nN]et[ao]|Meio|[Aa]vo)[A-Za-z ]*\.', elem)

            if(y):
                relacao = y.group()[:-1]
                if relacao not in dic:
                    dic[relacao] = 1
                else:
                    dic[relacao] += 1
    f.close()
    return dic
```

Figura 10: Frequência de relações

```
{'Tio Avo Paterno': 1, 'Tio Materno': 3, 'Irmao': 5, 'Tio Paterno': 1}
```

Figura 11: Exemplo de teste, Frequência de relações

Relacoes	Frequencia
Tio Avo Paterno	1
Tio Materno	3
Irmao	5
Tio Paterno	1

Figura 12: Tabela de teste, Frequência de relações

3.4 Registos em formato Json

Para imprimir em formato Json voltou a usar-se a função **split()** sobre as primeiras 20 linhas do ficheiro e criou-se um dicionário para cada linha que associa cada parâmetro do resultado anterior ao respetivo campo. Colocou-se os dicionários todos numa lista e gerou-se o ficheiro Json.Devido ao ficheiro Json ser extenso, apenas é mostrado parte do mesmo.

```
def to_json(ficheiro):

    f = open(ficheiro, 'r')

    campos = ["Numero", "Data", "Nome", "Pai", "Mae", "Informacao Adicional"]
    lista = []
    dic = {}
    for i in range(20):
        l = f.readline()
        x = re.split(r'::', l)
        for elem in range(len(campos)):
            dic[campos[elem]] = x[elem]
        lista.append(dic)
        dic = {}

    f.close()

    with open("output.json", 'w') as fp:
        json.dump(lista,fp, indent = 4)
```

Figura 13: Conversão para Json

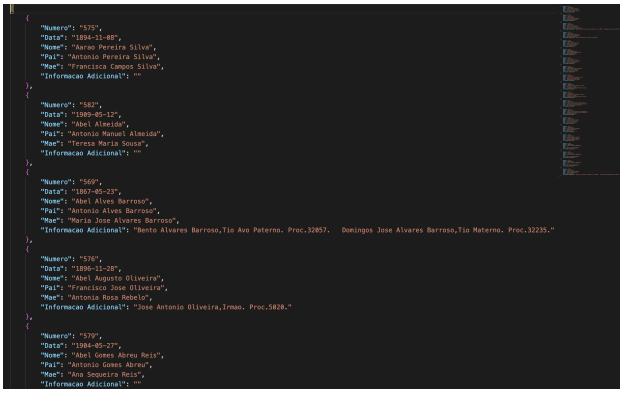

Figura 14: Exemplo de teste, Conversão para Json

4 Análise de resultados Processos.txt

Nesta seccção, irei mostrar os resultado das funções aplicadas ao ficheiro processos.txt. Em alguns casos, devido ao facto de o resultado ser muito extenso apenas irei mostrar parte do mesmo.

4.1 Frequência de processos por ano

Parte do dicionário e da tabela que asscociam a cada ano o número de processos.


```
{'1894': 47, '1909': 35, '1867': 45, '1896': 51, '1904': 46, '1901': 52, '1883': 31, '1900': 38, '1902': 62, '1
880': 54, '1889': 54, '1908': 40, '1869': 32, '1862': 32, '1906': 53, '1856': 65, '1875': 15, '1892': 46, '1733
': 976, '1778': 893, '1691': 960, '1730': 1000, '1899': 68, '1898': 58, '1877': 41, '1910': 25, '1881': 58, '19
07': 43, '1884': 38, '1879': 48, '1895': 49, '1897': 53, '1707': 106, '1689': 564, '1713': 233, '1824': 224, '1
703': 134, '1720': 154, '1890': 34, '1732': 1775, '1683': 134, '1863': 29, '1729': 39, '1694': 43, '1765': 8, '
1754': 279, '1690': 255, '1755': 282, '1823': 170, '1708': 141, '1757': 26, '1699': 90, '1759': 81, '1712': 66,
 '1687': 87, '1738': 167, '1717': 237, '1684': 221, '1704': 308, '1688': 104, '1888': 57, '1734': 732, '1786':
304, '1798': 43, '1773': 430, '1821': 260, '1822': 269, '1809': 249, '1722': 403, '1680': 150, '1695': 41, '172
8': 396, '1716': 214, '1849': 99, '1777': 914, '1851': 60, '1785': 682, '1857': 73, '1686': 169, '1784': 315, '
1780': 192, '1727': 145, '1788': 346, '1719': 336, '1847': 113, '1799': 86, '1829': 145, '1787': 606, '1805': 8
7, '1819': 201, '1844': 169, '1891': 49, '1731': 713, '1760': 239, '1741': 26, '1725': 253, '1802': 75, '1827':
 77, '1885': 8, '1807': 474, '1710': 172, '1692': 211, '1706': 102, '1858': 44, '1739': 59, '1826': 147, '1714'
: 288, '1762': 291, '1743': 51, '1724': 92, '1697': 26, '1852': 69, '1740': 12, '1855': 76, '1723': 118, '1859'
: 80, '1811': 181, '1817': 177, '1685': 215, '1905': 33, '1893': 56, '1865': 33, '1848': 103, '1911': 14, '1882
': 44, '1735': 118, '1871': 33, '1903': 18, '1850': 94, '1825': 138, '1843': 250, '1860': 71, '1812': 219, '184
6': 79, '1845': 111, '1701': 104, '1746': 45, '1868': 21, '1715': 83, '1803': 137, '1830': 157, '1761': 329, '1
766': 9, '1672': 17, '1876': 38, '1698': 148, '1726': 78, '1679': 74, '1750': 49, '1711': 58, '1810': 46, '1756
```


Figura 15: Frequência de processos por ano

Ano	Numero de processos
1894	47
1909	35
1867	45
1896	51
1904	46
1901	52
1883	31
1900	38
1902	62
1880	54
1889	54
1908	40
1869	32
1862	32
1906	53
1856	65
1875	15
1892	46
1733	976
1778	893
1691	960
1730	1000
1899	68
1898	58
1877	41
1910	25
1881	58
1907	43
1884	38
1879	48
1895	49
1897	53
1707	106
1689	564
1713	233
1824	224

Figura 16: Tabela, Frequência de processos por ano

4.2 Frequência de nomes próprios por século

Parte do dicionário e da tabela que associam a cada século os nomes próprios que ocorrem e a respetiva frequência.

Figura 17: Frequência de nomes próprios por século

Seculo	Primeiro Nome	Frequencia
19	Aarao	1
19	Antonio	3057
19	Francisca	257
19	Abel	3
19	Maria	3126
19	Francisco	1517
19	Antonia	354
19	Joao	2259
19	Cecilia	5
19	Abelardo	1
19	Jose	3035
19	Leopoldina	7
19	Abilio	7
19	Sebastiao	80
19	Teresa	437
19	Ana	1054
19	Flora	1
19	Acacio	3
19	Albina	14
19	Adelino	12
19	Custodia	271
19	Bernardo	172
19	Rosa	560
19	Manuel	2958
19	Rosalia	22
19	Joana	280
19	Adolfo	3
19	Carlota	14
19	Custodio	203
19	Emilia	25
19	Adriano	9
19	Miguel	120
19	Adriao	1
19	Bernardina	26

Figura 18: Tabela, Frequência de nomes próprios por século

4.3 Frequência de apelidos por século

Parte do dicionário e da tabela que associam a cada século os apelidos que ocorrem e a respetiva frequência.

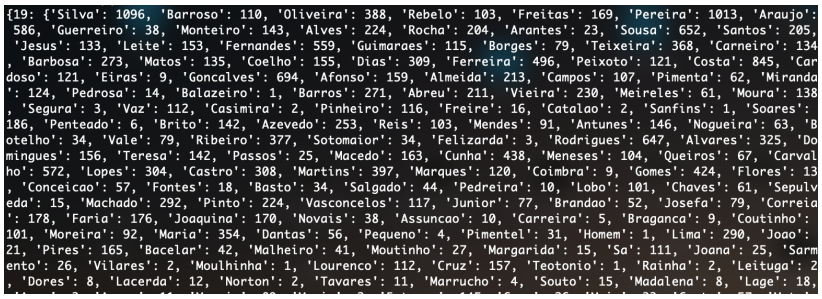

Figura 19: Frequência de apelidos por século

Seculo	Apelido	Frequencia
19	Silva	1096
19	Barroso	110
19	Oliveira	388
19	Rebelo	103
19	Freitas	169
19	Pereira	1013
19	Araujo	586
19	Guerreiro	38
19	Monteiro	143
19	Alves	224
19	Rocha	204
19	Arantes	23
19	Sousa	652
19	Santos	205
19	Jesus	133
19	Leite	153
19	Fernandes	559
19	Guimaraes	115
19	Borges	79
19	Teixeira	368
19	Carneiro	134
19	Barbosa	273
19	Matos	135
19	Coelho	155
19	Dias	309
19	Ferreira	496
19	Peixoto	121
19	Costa	845
19	Cardoso	121
19	Eiras	9
19	Goncalves	694
19	Afonso	159
19	Almeida	213
19	Campos	107
19	Pimenta	62

Figura 20: Tabela,Frequência de apelidos por século

4.4 Frequência de relações

Dicionário e tabela com o número de frequência das relações.

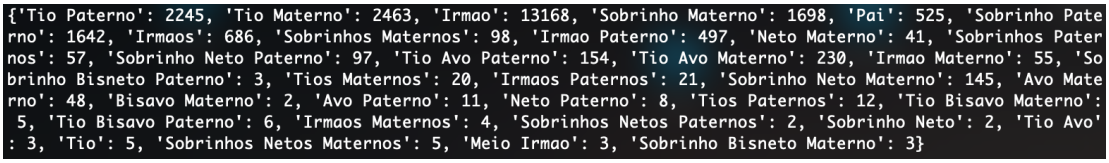

Figura 21: Frequência de relações

Relacoes	Frequencia
Tio Paterno	2245
Tio Materno	2463
Irmao	13168
Sobrinho Materno	1698
Pai	525
Sobrinho Paterno	1642
Irmaos	686
Sobrinhos Maternos	98
Irmao Paterno	497
Neto Materno	41
Sobrinhos Paternos	57
Sobrinho Neto Paterno	97
Tio Avo Paterno	154
Tio Avo Materno	230
Irmao Materno	55
Sobrinho Bisneto Paterno	3
Tios Maternos	20
Irmaos Paternos	21
Sobrinho Neto Materno	145
Avo Materno	48
Bisavo Materno	2
Avo Paterno	11
Neto Paterno	8
Tios Paternos	12
Tio Bisavo Materno	5
Tio Bisavo Paterno	6
Irmaos Maternos	4
Sobrinhos Netos Paternos	2
Sobrinho Neto	2
Tio Avo	3
Tio	5
Sobrinhos Netos Maternos	5
Meio Irmao	3
Sobrinho Bisneto Materno	3

Figura 22: Tabela, Frequência de relações

5 Conclusão

Com a realização deste trabalho, os objetivos esperados foram alcançados e a experiência na manipulação de expressões regulares aumentou.

A elaboração do projeto contribui para o desonlvimento de código em linguagem HTML e para o aumento da experiência relativamente a uma forma de armazenamento de dados e representação, o Json.

O desenvolvimento do relátorio em LaTex permitiu ainda aumentar o conhecimento sobre este sitema de edição de documentos.

6 Código dos ficheiros

6.1 Ficheiro func.py

O ficheiro **fun.py** é constituido pelo código mostrado na resolução das alíneas e pelas funções que transformam um dicionário em um ficheiro HTML seguintes:

```
def dic_html(headers,info,file):
    f = open("html/" + file, "w+")
    f.write("<!DOCTYPE html>\n")
    f.write("<html>\n")

    f.write("<style>\n")
    f.write("\ttable,td,th {\n \t\tborder: 1px solid black; \n")
    f.write("\t\tborder-collapse: collapse;\n \t\ttext-align: center;\n \t}\n")
    f.write("</style>\n")

    f.write("\t<table>\n")
    f.write("\t\t<tr>\n")
    # headers
    for i in range(0,len(headers)):
        f.write("\t\t\t<th>")
        f.write(headers[i])
        f.write("</th>\n")
    f.write("\t\t</tr>\n")

    for key,value in info.items():
        f.write("\t\t<tr>\n")
        f.write("\t\t\t<td>")
        f.write(key)
        f.write("</td>\n")
        f.write("\t\t\t<td>")
        f.write(str(value))
        f.write("</td>\n")
        f.write("\t\t</tr>\n")

    f.write("\t</table>\n")
    f.write("</html>")
    f.close()
```

Figura 23: Dicionário Python para HTML

```python
def dic_dic_html(headers,info,file):
    f = open("html/" + file, "w+")
    f.write("<!DOCTYPE html>\n")
    f.write("<html>\n")

    f.write("<style>\n")
    f.write("\ttable,td,th {\n \t\tborder: 1px solid black; \n")
    f.write("\t\tborder-collapse: collapse;\n \t\ttext-align: center;\n \t}\n")
    f.write("</style>\n")

    f.write("\t<table>\n")
    f.write("\t\t<tr>\n")
    # headers
    for i in range(0,len(headers)):
        f.write("\t\t\t<th>")
        f.write(headers[i])
        f.write("</th>\n")
    f.write("\t\t</tr>\n")

    for key,value in info.items():
        f.write("\t\t<tr>\n")
        for x,y in value.items():
            f.write("\t\t\t<td>")
            f.write(str(key))
            f.write("</td>\n")
            f.write("\t\t\t<td>")
            f.write(str(x))
            f.write("</td>\n")
            f.write("\t\t\t<td>")
            f.write(str(y))
            f.write("</td>\n")
            f.write("\t\t</tr>\n")

    f.write("\t</table>\n")
    f.write("</html>")
    f.close()
```

Figura 24: Aninhamento de dicionários Python para HTML

6.2 Ficheiro main.py

O ficheiro **main.py** é constituído pelo código relativo à implementação do menu e invocação das funções.

```
import func, os
def printMenu():

    print("\n \t\t\t\t\tMenu\n")
    print("\t[1] - Calcular a frequência de procsessos por ano.", end = "\n")
    print("\t[2] - Calcular a frequência de nomes próprios por século.", end = "\n")
    print("\t[3] - Calcular a frequência de apelidos por século.",end="\n")
    print("\t[4] - Calcular a frequência dos vários tipos de relação.",end="\n")
    print("\t[5] - Imprimir os 20 primeiros registos num novo ficheiro output em formato Json.", end="\n")
    print("\t[0] - Sair")


def init():


    printMenu()
    opc = input("Escolha uma opção: ")
    ficheiro = "processos.txt"


    while opc != 0:

        if opc == '1': #frequêncida de processos por ano
            print(func.freq_processos_ano(ficheiro))
            func.dic_html(["Ano", "Numero de processos"], func.freq_processos_ano(ficheiro), "freq_processos_ano.html")
            os.system("open -a \"Google Chrome\" html/freq_processos_ano.html")

        elif opc == '2': #frequencia de nomes proprios por século
            print(func.freq_nomes_proprios_sec(ficheiro))
            func.dic_dic_html(["Seculo", "Primeiro Nome", "Frequencia"], func.freq_nomes_proprios_sec(ficheiro), "freq_nomes_proprios_sec.html")
            os.system("open -a \"Google Chrome\" html/freq_nomes_proprios_sec.html")


        elif opc == '3': #frequencia de apelidos por século
            print(func.freq_apelidos_sec(ficheiro))
            func.dic_dic_html(["Seculo", "Apelido", "Frequencia"], func.freq_apelidos_sec(ficheiro), "freq_apelidos_sec.html")
            os.system("open -a \"Google Chrome\" html/freq_apelidos_sec.html")


        elif opc == '4': #frequencia de relaçõess
            print(func.freq_relacoes(ficheiro))
            func.dic_html(["Relacoes", "Frequencia"], func.freq_relacoes(ficheiro), "freq_relacoes.html")
            os.system("open -a \"Google Chrome\" html/freq_relacoes.html")

        elif opc == '5':
            print(func.to_json(ficheiro))
            os.system("open output.json")



        elif opc == '0':
            print("Fim")
            break

        print()
        printMenu()
        opc = input("Escolha uma opção: ")

init()
```

Figura 25: Código ficheiro main.py