INSTITUTO POLITÉCNICO
DO CÁVADO E DO AVE
ESCOLA SUPERIOR DE TECNOLOGIA

# Relatório do 1º trabalho prático

## Tema A

## Processamento de Linguagens

Licenciatura de Engenharia de Sistemas Informáticos (pós-laboral)

Luís Esteves - 16960

Sérgio Ribeiro - 18858

## 1. Objetivo do trabalho

O grupo escolheu o tema A. Este tema tem como proposta realizar um programa que consiga ler dicionários que tem a seguinte formatação:

```
#definicao = Fio que interrompe o circuito eléctrico.
PT = Fusível
+genero = Masculino
+numero = Singular
+exemplo = Saltaram os fusíveis.
EN = Fuse
DE = Sicherung
+genero = Feminino
+numero = Singular
FR = Plomb
+genero = Masculino
+numero = Singular

definicao = Reservatório.
PT = Depósito de gasolina
+genero = Masculino
+numero = Singular
EN = Petrol tank
DE = Benzintank
+genero = Masculino
+numero = Singular
```

*Figura 1: Exemplo*

Para além de ler, é necessário converter este ficheiro num Html e num LaTex. Finalmente, depois desta conversão, fica ao encargo de cada grupo desenvolver e ter ideias sobre como explorar a linguagem Python e a manipulação de expressões regulares.

## 2. Desenvolvimento do Projeto

Ao desenvolver este projeto, para além da conversão do ficheiro em Html e num LaTex, construímos funções que possibilitam o utilizador:

1 - Contar as palavras do dicionário;

2 - Encontrar uma palavra que o utilizador deseja encontrar no dicionário.

3 - Ver quantas palavras começam por uma determinada letra.

4 - Demonstrar os atributos de uma dada palavra.

## 2.1 Expressões regulares utilizadas

1-> r"[A-Z]+(\-[A-Z]+)*[ ]*.[ ]*([A-Z]*[a-z]+[ ]*|.+)+"
2-> r"\+.+[ ]*=[ ]*(\,?[ ]*.[ ]*)+"
3-> r"[a-z][ ]*.[ ]*.+"
4-> r"(\#[ ]*.+)|[ ]*.+"

A primeira expressão foi feita para reconhecer frases com seguinte formato “EN = house”, a segunda para este “+número=singular”, a terceira para “definição=reservatório de agua” e a quarta para “#defenicao=blablabla…”

## 2.2 Contar as palavras do dicionário

```
#Código da Opção 1 (Contar palavras de um dicionário)
def opc1(P):
    Linhas_mortas = 0
    for elemento in P:
        if elemento == '0':
            Linhas_mortas = Linhas_mortas + 1
    print(f"Total de palavras: {len(P) - Linhas_mortas}")
    print("----------------------------------------")
```

*Figura 2: Código da Função que conta as palavras do dicionário*

Para contar as palavras (quando referimos ‘palavras’, estamos a falar da segunda parte de linhas como esta “EN = house”) que os dicionários contêm, primeiro é necessário ler o próprio ficheiro. Depois disso, com a seguinte expressão regular:

r"[A-Z]+(\-[A-Z]+)*[ ]*.[ ]*([A-Z]*[a-z]+[ ]*|.+)+"

Depois de reconhecer as linhas em que as palavras se encontram, fazemos o split do token (feito da seguinte maneira):

```python
#Faz split dos tokens encontrados.
def splitF(palavra, tipo):
    if tipo == 0:
        for letra in palavra:
            if letra == '=':
                p = palavra.split("=")
                palavras_dictionary = {"Linguagem": p[0].strip(), "Palavra": p[1].strip()}
                return palavras_dictionary
            if letra == ':':
                p = palavra.split(":")
                palavras_dictionary = {"Linguagem": p[0].strip(), "Palavra": p[1].strip()}
                return palavras_dictionary
            if letra == ',':
                p = palavra.split(",")
                palavras_dictionary = {"Linguagem": p[0].strip(), "Palavra": p[1].strip()}
                return palavras_dictionary
    elif tipo == 1:
        for letra in palavra:
            if letra == '=':
                p = palavra.split("=")
                palavras_dictionary = {"Caracteristica": p[0].strip(), "Palavra": p[1].strip()}
                return palavras_dictionary
            if letra == ':':
                p = palavra.split(":")
                palavras_dictionary = {"Caracteristica": p[0].strip(), "Palavra": p[1].strip()}
```

*Figura 3: Uma parte da função realiza o split dos tokens*

Guardamos cada uma das partes numa lista. Por fim, contamos o número de elementos que são diferentes de nulo e sabemos então quantas palavras existem no dicionário.

## 2.3 Encontrar uma palavra

```python
#Código da Opção 2 (Encontrar uma palavra)
def opc2(P):
    paux = 0
    pnaux = 0
    print("\n\n-----------------------------------------")
    pal = input("\nDigite a palavra que deseja procurar:")
    try:
        for elemento in P:
            paux = paux + 1
            if elemento == pal:
                print(f"Palavra encontrada: {elemento}")
                print("-----------------------------------------")
            else:
                pnaux = pnaux + 1
        if pnaux == paux:
            print("Palavra não encontrada!")
            print("-----------------------------------------")
    except:
        print("Erro na digitação. Repita por favor.")
        print("-----------------------------------------")
```

*Figura 4: Função que permite encontrar uma dada palavra*

Realizamos o mesmo processo que em cima, só que com a seguinte expressão regular:

r"\+.+[ ]*=[ ]*(\,?[ ]*.[ ]*)+”

Depois de ter a lista construída, realizamos um ciclo “for” para encontrar a palavra que o utilizador deseja encontrar.

## 2.4 Ver quantas palavras começam com uma dada palavra

```
#Código da Opção 3 (Ver quantas palavras começam por uma determinada letra)
def ocp3(P):
    numerolista = 0
    totalPalavras = 0
    l = input("Quantas palavras começam pela letra-> ")
    try:
        for elemento in P:
            numerolista = numerolista + 1
            comecoPalavra = 0
            for letra in elemento:
                comecoPalavra = comecoPalavra + 1
                if letra == l and comecoPalavra == 1:
                    print(f"Palavra -> '{P[numerolista - 1]}'")
                    totalPalavras = totalPalavras + 1
        print("\n-----------------------------------")
        print(f"Palavras encontradas que começam pela letra '{l}': {totalPalavras}")
    except:
        print("Erro! Por favor insira um valor correcto.")
```

*Figura 5: Função*

O utilizador insere a letra que deseja, e a partir realiza-se dois ciclos “for” a correr a lista com as palavras todas. O primeiro correr a lista das palavras e o segundo para ver se a letra começa pela mesma letra que o utilizador inseriu. Caso seja, então imprimimos a mesma palavra na consola. Quando o ciclo “for” termina, imprimimos então quantas palavras existem no dicionário com a letra que o utilizador inseriu.

## 2.5 Ver os atributos de uma dada palavra

```python
#Código da Opção 4(Demonstrar os atributos de uma dada palavra)
def ocp4(p, a, a2, d, d2):
    i = numerolista = vez = linhaEncontrada = linhaAux = nLista = 0
    listaEncontrados = [0,0,0,0,0,0,0,0]
    palavra = input("Digite a palavra quer ver os atributos->")
    palavra2 = palavra.strip()
    try:
        for elemento in p:
            numerolista = numerolista + 1
            if (elemento == palavra2 and linhaEncontrada == 0) or (elemento == palavra2 and listaEncontrados[nLista] == 0):
                    linhaEncontrada = numerolista-1
                    linhaAux = linhaEncontrada
                    listaEncontrados[nLista] = linhaAux
                    nLista = nLista + 1
        for elemento in a:
            if linhaEncontrada == i and listaEncontrados[1] == 0:
                if d[linhaEncontrada-1] != '0':
                    print(f"{d[linhaEncontrada-1]}:{d2[linhaEncontrada-1]}")
                    vez = 1
                linhaAux = linhaAux + 1
                while d[linhaAux] != '0' or a[linhaAux] != '0':
                    if d[linhaAux] != '0':
                        print(f"{d[linhaAux]}:{d2[linhaAux]}")
                        vez = 1
                    if a[linhaAux] != '0':
                        print(f"{a[linhaAux]}:{a2[linhaAux]}")
                        vez = 1
```

*Figura 6: Função que permite ao utilizador ver os atributos de uma palavra*

Esta função é um pouco mais complicada. Primeiro verificamos se a palavra existe na lista que guarda todas as palavras do dicionário. Depois, no caso de a palavra existir apenas só uma vez, vamos percorrer a lista que guarda todos os atributos de todas as palavras, e quando chegarmos ao índice igual à linha em que encontramos a palavra que o utilizador inseriu, começamos a imprimir na consola os atributos da mesma.

No caso da palavra se repetir no dicionário, fazemos o mesmo, porém percorremos a lista que guarda o índice das palavras encontradas e repetimos o processo de cima as vezes necessárias.

## 3. Conversão do ficheiro

Para realizar a conversão do dicionário, temos as seguintes funções:

```
#Converte o ficheiro para HTML.
def convert(filename):
    contents = open(filename, "r")
    with open("suleiman.html", "w") as e:
        for lines in contents.readlines():
            if lines != '\n': e.write(lines + "<br>\n")

def convert2(filename):
    docutils.core.publish_file(
        source_path=filename,
        destination_path="output.tex",
        writer_name="latex")
```

*Figura 7: Funções que fazem as conversões para Htmle para Latex, respetivamente*

Com os seguintes outputs:

PT = cantor
EN-GB = singer
DE = SÃ¤nger
PT = piano
EN-GB = piano
DE = Klavier
PT = concerto
EN-GB = concert
DE = Konzert
PT = banda
EN-GB = band
DE = Bande
PT = danÃ§arino
EN-GB = dancer
DE = TÃ¤nzer
PT = harmonia
EN-GB = harmony
DE = Harmonie
PT = flauta
EN-GB = flute
DE = FlÃ¶te
PT = notas
EN-GB = notes
DE = Noten
PT = canÃ§Ã£o
EN-GB = song
DE = Lied
PT = danÃ§a
EN-GB = dance
DE = Tanz
PT = solo
EN-GB = solo
DE = Solo
PT = coro
EN-GB = choir
DE = Chor

*Figura 8: Ficheiro convertido em Html*

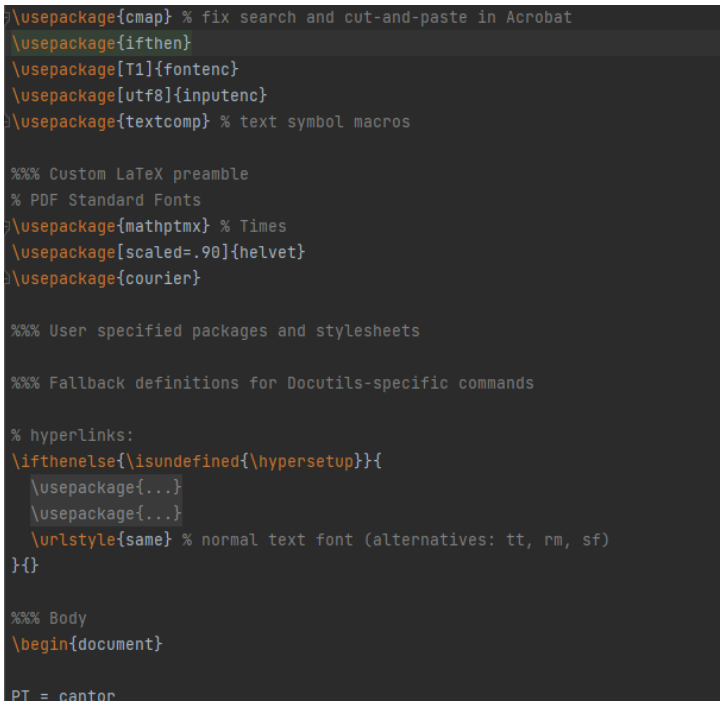


*Figura 9: Ficheiro convertido em LaTex*

## 4. Dificuldades encontradas

Para além da conversão dos ficheiros em Html e em Latex, que se revelou uma tarefa complicada devido ao vasto, porém fraco material que existe na internet, especialmente para o segundo tipo de conversão. Para além disso, também foi complicado guardar os tokens de uma maneira eficiente e que desse para manipular da maneira desejada.