SPLN

MI-EI

Teste, 18 de Janeiro de 2019

Questão 1. Escreva da forma mais simples e elegante que conseguir:

(a) Uma função Python que, dada uma lista de números, dê o valor absoluto da maior diferença entre dois elementos dessa lista:

```
max_diff([1,1,3,5,2]) = 4
```

(b) Uma função Python que, dada uma string, construa um dicionário que conta as ocorrências de cada carácter da string:

```
count_char_occur('TESTE DE SPLN') = {
  'T': 2,
  'E': 3,
  'S': 2,
  ' ': 2,
  'D': 1,
  'P': 1,
  'L': 1,
  'N': 1
}
```

Questão 2. Implemente em Python um filtro UNIX que remova de um texto quebras de linha (caracteres `\n`) e quebra de palavra (translineações), preservando quebras de paráfrafo (dois `\n` seguidos):

```
str = "Ele mesmo costumava dizer que\n era simplesmente um egoísta:  mas\n nunca,
como agora na velhice, as\n generosidades do seu coração ti-\n nham sido tão
profundas e largas.\n\n Parte do seu rendimento ia-se-\n -lhe por entre os dedos,
espar-\n samente, numa caridade enterne-\n cida."

fix_lines(str) = "Ele mesmo costumava dizer que era simplesmente um egoísta:  mas
nunca, como agora na velhice, as generosidades do seu coração tinham sido tão
profundas e largas.\n\n Parte do seu rendimento ia-se-lhe por entre os dedos,
esparsamente, numa caridade enternecida."
```

Questão 3. Suponha que temos um texto T_{min} cujas maiúsculas (nomes próprios, siglas, inícios de frase) foram inadvertidamente todas convertidas para minúsculas.

(a) Implemente uma função `fix_sent_start` que receba como argumento uma string com T_{min} e corrija as maiúsculas em início de frase.

(b) Suponha agora que dispõe de um texto grande Tg com frases que incluem nomes próprios, siglas, etc, e que pretendemos usá-lo para corrigir as outras minúsculas indevidas em T_{min}.

Implemente uma função que recebe como argumento Tg e devolve uma estrutura com informação relevante sobre as ocorrências de palavras em maiúscula e minúsculas para fazer essa correção. Descreva detalhadamente a informação contida na estrutura e a sua relevância para a recuperação de maiúsculas.

Questão 4. Suponha que dispomos de um tabela de polaridade de palavra que a cada palavra associe um número no intervalo [-1:+1], sendo -1 associado a sentimento muito negativo, 0 a neutro e +1 francamente positivo. Suponha também que existe uma coleção de notícias (textos).

Escreva uma função Python que receba (i) uma string com um nome (ex: "José Mourinho"), (ii) um dicionário com a polaridade das palavras, e (iii) uma lista com as notícias, e:

(a) procure nas notícias as ocorrências do nome e extraia uma janela de 10 palavras à sua volta,

(b) calcule o índice de popularidade como sendo o somatório das polaridades das palavras a dividir pelo número de ocorrências do nome.

Questão 5. Suponha que dispomos de dois textos, `t1` e `t2`, que são iguais mas com algumas palavras escritas com grafia diferente.

```
t1 = "As inundações provocaram graves damnos na pharmácia. [...]"
t2 = "As inundações provocaram graves danos na farmácia. [...]"
```

Implemente uma função `find_corresp` que receba essas duas strings e retorne um dicionário com as correspondências de grafia:

```
find_corresp(t1, t2) = {
  'damnos': 'danos',
  'pharmácia': 'farmácia'
}
```

Assuma que cada palavra em `t1` corresponde a uma e uma só palavra em `t2`, e que todas as ocorrências de qualquer palavra em `t1` têm sempre a mesma palavra correspondente em `t2`.

Questão 6. Pretende-se implementar um gerador de *chat bots*. Neste gerador, um *bot* é uma função que recebe uma lista de tuplos, em que cada tuplo é composto por um padrão (expressão regular) e uma lista de acções (identificadores de funções, funções *lambda* ou constantes), às quais são passados os grupos de captura das expressões regulares:

```
accoes = [
  (r'és um (\w+)', [
     lambda x: return f'{x} és tu!,
     lambda x: return f'Tu é que és {x}',
     'Quem diz é quem é!'
  ]),
  (r'.', ['Não percebi, fala direito!'])
]
```

O *bot* recebe ainda uma string, e devolve a resposta a essa string:

```
bot_responde(accoes, 'Carro em inglês') = 'car'
bot_responde(accoes, 'És um burro') = 'burro és tu!'
```

(a) Implemente o padrão e a acção correspondentes ao primeiro exemplo de resposta do *bot* ("`Carro em inglês`"). Assuma que existe um dicionário `dic_pt_en` definido globalmente cujas chaves são palavras em Português e os valores as respectivas traduções em Inglês.

(b) Implemente a função `bot_responde` que recebe uma lista de acções e uma string, procura qual o primeiro padrão que corresponde à string e dispara a acção correspondente (ou uma das acções, escolhida aleatoriamente). Pode tirar partido da função `callable(objecto)` que devolve `True` se o seu argumento for invocável, e `False` caso contrário.