SPLN

MI-EI

Teste, 3 de Junho de 2019

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

(a) Uma função Python que, dada uma lista de números inteiros, retorna a soma do número de dígitos desses números:

```
count_digits([1, 5, 14, 28, 1000]) = 10
```

(b) Uma função Python que, dada uma String, verifica se é um palíndromo ou expressão palíndroma: *"palavra, grupo de palavras ou verso em que o sentido é o mesmo, quer se leia da esquerda para a direita quer da direita para a esquerda"*.[1] Para este efeito, ignore os espaços, pontuação, capitalização e caracteres acentuados (pode usar a função `unidecode` do módulo `unidecode` que recebe uma String e a devolve sem acentuação).

```
is_palindrome("Olé! Maracujá, caju, caramelo.") # True
is_palindrome("Scripting em PLN")               # False
```

Questão 2. Implemente em Python um filtro UNIX que receba um texto etiquetado em formato `palavra/tag` e imprima, para cada palavra do texto, a palavra e o número total de ocorrências, a(s) tag(s) correspondente(s) e o número de vezes que a palavra foi etiquetada com cada tag.

```
$ cat text.txt
"Eu/PROPESS nunca/ADV almoço/V à/PREP+ART hora/N do/KS almoço/N ./."


$ words_tags text.txt
almoço (2):  V (1), N (1)
Eu (1):  PROPESS (1)
nunca (1):  ADV (1)
à (1):  PREP+ART (1)
hora (1):  N (1)
do (1):  KS (1)
.  (1):  .  (1)
```

Questão 3. Um ficheiro de texto foi obtido através do OCR (reconhecimento ótico de caracteres) de um livro. Em várias das palavras, um dos caracteres não foi reconhecido, tendo sido representado pelo símbolo %. Pretende-se recuperar as palavras originais, usando N-gramas para o efeito.

(a) Implemente uma funcao `calc_trigrams` que, dada um texto grande calcula as ocorrências de trigramas de caracteres nesse texto.

(b) Implemente uma função `fix_word` que recebe uma palavra com um caracter ilegível (representado por %) e encontra a palavra correspondente mais provável.

```
fix\_word('univer%idade') # 'universidade'
```

Questão 4. Uma universidade pretende implementar um sistema de deteção de plágio em teses de mestrado e doutoramento. Este sistema deverá pesquisar semelhanças com artigos ou teses antigas. No caso de não encontrar nada, o sistema deve notificar o júri com `NÃO_PLÁGIO`. Caso encontre uma obra bastante parecida, o sistema deve enviar para o júri um aviso de potencial `PLÁGIO`, e o link da obra plagiada. O júri depois confirmará manualmente se houve efectivamente plágio.

Duas ferramentas concorrentes, A e B, foram avaliadas. Para cada ferramenta foi criada uma colecão de textos (C_A e C_B) classificados manualmente como PLÁGIO ou NÃO_PLÁGIO. Cada ferramenta

Ferramenta A		Classificação automática	
		PLÁGIO	**NÃO_PLÁGIO**
Classificação manual	**PLÁGIO**	5	0
	NÃO_PLÁGIO	5	90

Ferramenta B		Classificação automática	
		PLÁGIO	**NÃO_PLÁGIO**
Classificação manual	**PLÁGIO**	20	10
	NÃO_PLÁGIO	0	40

foi depois usada para classificar automaticamente os documentos da coleção respectiva. Os valores esperados e obtidos estão representados nas tabelas em baixo.

(a) Calcule os valores de *precision* (precisão) e *recall* (cobertura) para cada ferramenta.

(b) Qual deveria ser a ferramenta escolhida para o cenário descrito? Justifique.

(c) Descreva como implementaria uma ferramenta deste género.

Questão 5. Pretende-se construir um sistema para gerar respostas a partir de uma estrutura Python de acordo com os seguintes casos:

```
g( "frase" )         = frase
g( [f1,f2,... fn] )  : sorteia um dos elementos e gera-o
g( (p1, p2, ... pn)) : concatena as partes
g( lambda a: g(a))   : invoca passando Estado como parâmetro
```

```
Estado={"nome": "Joaquim"}

g(
   (
     [ ( "meu caro",
         [ "amigo,",
           lambda x: x["nome"]
         ]),
       "querido interlocutor"
     ],
     [ "cala-te que já lá vai o tempo em que os animais falavam.",
       "estou profundamente abismado com o que me dizes."],
   )
 )
```

podendo gerar frases como:
meu caro Joaquim estou profundamento abismado com o que me dizes.

Implemente a função `g(S)`. Pode tirar partido da função `isinstance(var, class)` que permite verificar se o primeiro argumento é instância do segundo, e da função `callable(obj)` que permite determinar se o argumento é invocável.

Questão 6. Pretende-se construir uma aplicação de linha de comandos para consultar a previsão de meteorologia na página do Instituto Português do Mar e da Atmosfera.

(a) Assuma que existe uma variável `weather` que contém dados da previsão para esta semana, no seguinte formato:

[1] `https://dicionario.priberam.org/palindromo`

```
weather = [{
    date: '2 Dom',
    prev_txt: 'Céu nublado por nuvens altas',
    temp_min: 13, temp_max: 28, uv: 8
  }, {
    date: '3 Seg',
    prev_txt: 'Céu pouco nublado',
    temp_min: 11, temp_max: 27, uv: 9,
  }, {
    date: '4 Qua',
    prev_txt: 'Céu limpo',
    temp_min: 9, temp_max: 31,
  }
]
```

Implemente uma função `conselhos_da_avo()` que recebe a variável `weather` e imprime "`Leva um casaco!`" se a temperatura mínima de hoje estiver abaixo dos 15°C, e "`Sai do sol que faz mal!`" se o índice UV estiver acima de 7. O módulo `datetime` exporta uma variável `date` que pode ser usada para obter o dia do mês em que estamos: `date.today().day`.

(b) Supondo que o URL para consultar a previsão meteorológica para Barcelos é `http://www.ipma.pt/Braga&Barcelos`, implemente a função `get_weather`, que:

- recebe como parâmetros o distrito e o concelho
- faz *scrapping* da página correspondente
- extrai a informação relevante usando a biblioteca `BeautifulSoup`
- devolve a informação extraída no mesmo formato da variável `weather` da alínea anterior.

Assuma que o HTML resultante do scrapping da página tem o seguinte formato:

```
<html>
  <head>[...]</head>
  <body>
    <div id="Container">
      <div id="Main">
        <div id="LeftCol_container">[...]</div>
        <div id="Bread_container">[...]</div>
        <div id="CRcol_container">
          <h2 class="bckg_blue local-header">Braga, Barcelos</h2>
          <div id="weekly">
            <div id="2" class="weekly-column active">
              <div  class="date">2 Dom</div>
              <img  class="weatherImg" src="[...]" title="Céu nublado por nuvens altas">
              <span class="tempMin">14º</span>
              <span class="tempMax">20º</span>
              <img  class="windImg d135" src="[...]" alt="Vento: Moderado de NW"
                    title="Vento: Moderado de NW">
              <div  class="windDir">NW</div>
              <div  class="precProb" alt="Probabilidade precipitação: 10%"
                    title="Probabilidade precipitação: 10%">[...]</div>
              <div  class="wrapper">[...]</div>
              <img  class="iuvImg" src="[...]" alt="I. Ultravioleta: 8" title="IUV: 8">
            </div>
            <div id="3" class="weekly-column">
              <div  class="date">3 Seg</div>
              <img  class="weatherImg" src="[...]" title="Céu pouco nublado">
              <span class="tempMin">12º</span>
              <span class="tempMax">25º</span>
              <img  class="windImg d180" src="[...]" alt="Vento: Moderado de N"
                    title="Vento: Moderado de N">
              <div  class="windDir">N</div>
              <div  class="precProb" alt="Probabilidade precipitação: 0%"
                    title="Probabilidade precipitação: 0%">[...]</div>
              <div  class="wrapper">[...]</div>
              <img  class="iuvImg" src="[...]" alt="I. Ultravioleta: 9" title="IUV: 9">
            </div>
          </div>
```

```
        </div>
        <div id="BotsCont_container">[...]</div>
      </div>
    </div>
  </body>
</html>
```