Prova Final de Programação 1 (IF968)

Fernando Castor
Centro de Informática
Universidade Federal de Pernambuco

12 de dezembro de 2016

1. (3,0 ptos.) De acordo com a Wikipédia[1], em estatística o desvio padrão (σ) é uma medida usada para quantificar a quantidade de variação ou dispersão de um conjunto de valores. Um desvio padrão baixo indica que os pontos têm uma tendência a estar próximos à média do conjunto e um desvio padrão alto indica que os pontos de dados estão espalhados ao longo de uma ampla faixa de valores. Se considerarmos que todas as amostras $x_0, x_1, x_2, ..., x_{(n-1)}$ são igualmente prováveis, o desvio padrão é dado pela seguinte fórmula:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{1}$$

onde a média aritmética μ é dada por

$$\mu = \sum_{i=1}^{N} x_i \tag{2}$$

Tendo isso tudo em vista, construa uma função `std_dev` que calcula o desvio padrão de uma lista de números. Nesta questão é proibido usar a função `sum`. Para calcular a raiz quadrada de um número use a função `sqrt`. Para usá-la, basta incluir `import math` no início do seu arquivo `.py`. Para usar a função `sqrt`, faça como no exemplo a seguir:

```
>>> math.sqrt(9)
3
```

Exemplos de resultados da função `std_dev`:

```
>>> std_dev([1,2,1,2,1,2,1,2,1,2])
0.5
>>> std_dev([6,6,6,6,6,6,6])
0.0
>>> std_dev([1,2,3,4,5])
1.4142135623730951
>>> std_dev([1.2, 4.5, 2.6, 7.35, 10, 5.4, 12.3, 3.75])
3.529938915902087
```

2. **(7,0 ptos.)** Um tokenizador é um programa que, dados um string a ser tokenizado e um string contendo caracteres separadores, quebra o primeiro string em palavras com base nos separadores, devolvendo uma lista de palavras (strings) como resultado. Construa uma função chamada `tokenize` que funciona como um tokenizador. Na resolução desta questão, é proibido usar laços (`while`, `for`). Também é proibido usar a função `split` e o operador `in`. Tenha em mente que strings em Python são imutáveis. Logo, não dá para usar a função `pop` em um string. Por outro lado, strings podem ser indexados como listas, seu tamanho pode ser obtido pela função `len`, é possível criar cópias de um string ou de pedaços dele e strings podem ser concatenados como listas:

```
>>> a = "abc"
>>> a[0]
```

[1] https://en.wikipedia.org/wiki/Standard_deviation

```
'a'
>>> len(a)
3
>>> b = "def"
>>> c = a + b
>>> c
'abcdef'
>>> c[1:4]
'bcd'
```

A resolução deve estar organizada da seguinte maneira:

(a) (1,0 pto.) uma função para determinar se um string contendo apenas um caractere é um elemento de um segundo string.

```
>>> elem('d', 'abcd')
True

>>> elem('x', 'abcd')
False
```

(b) (2,0 ptos.) uma função para, dados um string de texto e um string contendo separadores, obter a primeira palavra que aparece no primeiro string, por exemplo:

```
>>> pegaPalavra("abc,def,ghi", ",")
'abc'

>>> pegaPalavra("Saitama foi até a lua, mas voltou! Sensacional.", ",! ")
'Saitama'

>>> pegaPalavra("(81)123-456-789", "()-")
'81'
```

No segundo exemplo acima, note que o caractere espaço em branco também aparece como um dos separadores (segundo parâmetro passado para `tokenize`).

(c) (1,0 pto.) uma função para, dados um string de texto e um string contendo separadores, obter toda a parte desse primeiro string que aparece após a primeira palavra. Para o primeiro exemplo do item anterior, essa função devolveria `",def,ghi"`.

(d) (1,0 pto.) uma função para, dados um string de texto e um string contendo separadores, obter toda a parte desse primeiro string que aparece após a primeira sequência de separadores, por exemplo:

```
>>> tiraSeparadores("(81)123-456-789", "()-")
'81)123-456-789'

>>> tiraSeparadores("      ok, meu!", " ")
'ok, meu!'
```

(e) (2,0 ptos.) a função **tokenize** propriamente dita, como o exemplo a seguir:

```
>>> tokenize("abc,def,ghi", ",")
['abc', 'def', 'ghi']

>>> tokenize("Saitama foi até a lua, mas voltou! Sensacional.", ",! ")
['Saitama', 'foi', 'até', 'a', 'lua', 'mas', 'voltou', 'Sensacional.']

>>> tokenize("(81)123-456-789", "()-")
['81', '123', '456', '789']
```