Universidade Federal do Espírito Santo

Departamento de Computação e Eletrônica - CEUNES
Programação Funcional / Programação I
Prof. Oberlan Romão

Lista de exercícios 4

1. Defina uma função que:

   a. Retorna `True` se todos os elementos da lista forem números (inteiros ou reais) e `False`, caso contrário.

   b. Retorna a média dos elementos de uma lista;

   c. Retorna o menor e o maior elemento de uma lista de notas, sem utilizar as funções integradas `max` e `min`. Suponha que as notas são entre 0.0 e 10.0;

   d. Retorna o reverso da lista. Sua função deve ser recursiva, não podendo utilizar o `step` como sendo -1, por exemplo, `lista[::-1]`;

   e. Recebe uma lista e um elemento x e retorna quantas vezes x aparece na lista;

   f. Receba um inteiro n e retorna uma lista contendo todos os seus divisores;

2. Defina uma função que retorne uma lista contendo os n primeiros números primos. Dica: crie uma função auxiliar que retorna `True` se um número é primo e `False`, caso contrário;

3. Faça uma função que receba duas listas (`l1` e `l2`) e retorne uma lista com a união de `l1` e `l2` de forma que a nova lista não contenha elementos repetidos. Por exemplo, se `l1 = [1, 2, 3, 3]` e `l2 = [1, 5, 3]`, a união de `l1` e `l2` deve ser `[1, 2, 3, 5]` (a ordem dos elementos na lista não importa). Dica: use o operador `in`;

4. Defina uma função que receba uma lista `l1` e retorne outra lista de forma que elementos repetidos em `l1` apareça apenas uma vez na nova lista. Por exemplo, se `l1 = [1, 2, 3, 3, 2]`, sua função deve retornar `[1, 2, 3]`.

5. Faça uma função que receba duas listas e retorne uma lista com a interseção dessas listas. Por exemplo, se `l1 = [1, 2, 3, 3]` e `l2 = [1, 5, 3]`, a interseção de `l1` e `l2` deve ser `[1, 3]`. Note que a lista resultante não pode conter elementos repetidos;

6. Faça uma função que receba uma lista e retorna `True` se a lista estiver ordenada de forma crescente e `False`, caso contrário.

7. Faça um programa que peça para o usuário digitar valores inteiros entre 0 e 9, uma entrada fora dessa faixa indica o fim da leitura. Em seguida, seu programa deve imprimir a quantidade de números digitados e a frequência (quantidade de vezes) que cada número foi digitado. Veja um exemplo:

```
Digite os valores (um número <0 ou >9 indica o fim da leitura):
1
1
1
5
1
0
0
3
2
5
5
1
5
6
7
-1

Numeros digitados: 15
Frequencia de cada numero
0: 2
1: 5
2: 1
3: 1
4: 0
5: 4
6: 1
7: 1
8: 0
9: 0
```

8. Elabore um programa que peça ao usuário para inserir a temperatura dos últimos n dias ($1 \leq n \leq 100$) e calcule a temperatura média (m) e o desvio padrão (dp) considerando as informações inseridas pelo usuário. Seu programa deve, obrigatoriamente, ter três funções (com exceção da `main()`): uma para fazer a leitura das temperaturas, uma para calcular a média e outra para calcular o desvio padrão. A primeira recebe o valor de n e faz a leitura dos dados, a segunda recebe uma lista com as temperaturas e retorna a média dos valores, a terceira calcula e retorna o desvio padrão.

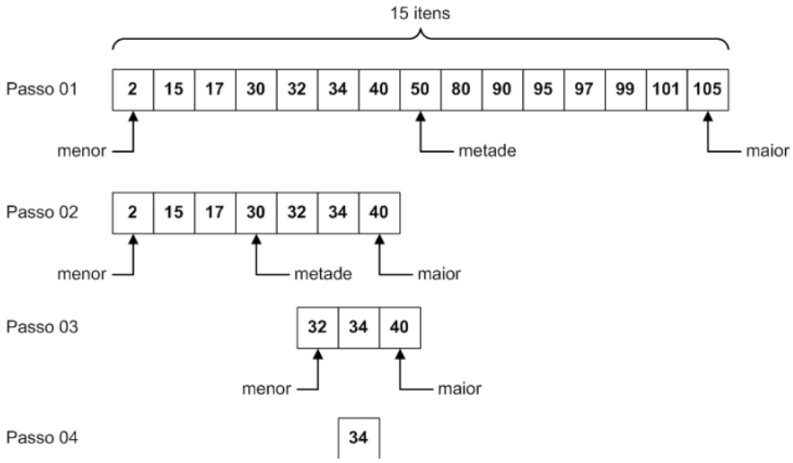
   Obs.: Supondo que a média de um conjunto de n valores seja m, o desvio padrão é dado pela expressão abaixo (onde t_i é o valor de temperatura no dia i).

$$dp = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(t_i - m)^2} = \sqrt{\frac{1}{n}\left((t_1 - m)^2 + (t_2 - m)^2 + (t_3 - m)^2 + \cdots + (t_n - m)^2\right)}$$

9. Um algoritmo de busca muito utilizado quando os elementos de uma lista estão ordenados é a *Busca Binária*. A ideia do algoritmo é testar o elemento procurado com o valor do elemento armazenado no meio da lista. Se o elemento buscado for menor que o elemento do meio, pode-se concluir que, se o elemento estiver presente no vetor, ele estará na primeira parte do vetor; se for maior, estará na segunda parte do vetor; se for igual, o elemento no vetor foi encontrado. Se for concluído que o elemento está em uma das partes da lista, o procedimento é repetido considerando apenas a parte

que restou: compara-se o elemento pesquisado com o elemento armazenado no meio dessa parte. Este procedimento é continuamente repetido, subdividindo a parte de interesse, até o elemento ser encontrado ou se chegar a uma parte da lista com tamanho zero.

A figura abaixo exemplifica a busca binária em uma lista ordenada. Neste caso, está sendo pesquisado o elemento 34.



Implemente uma função (`buscaBinariaR`) que faça uma Busca Binária recursiva em uma lista . Se o elemento procurado (`x`) estiver na lista, a função deve retornar a posição onde ele se encontra, caso contrário, ela deve retornar `None`. Sua função deve obedecer ao cabeçalho apresentado abaixo.

```
def buscaBinaria(L, x):
  """
  Função utilizada apenas para facilitar a chamada da função buscaBinariaR
  """
  return buscaBinariaR(L, x, 0, len(L)-1)

def buscaBinariaR(L, x, ini, fim):
  #implemente aqui
```

10. Faça um programa que contenha duas função: `ehVogal` e `contaVogal`. A primeira recebe um caractere e retorna `True` se o caractere é uma vogal e `False`, caso contrário. A segunda função, (`contaVogal`), receba uma string e retorne a quantidade de vogais contidos na string, ela deve obrigatoriamente usar a função `ehVogal`. Implemente, também, uma função `main()` que lê uma string e utiliza a sua função `contaVogal` para imprimir o número de vogais na string.

11. Considerando as funções do exercício anterior e supondo que uma string contenha apenas letras (minúsculas ou minúsculas), implemente uma função `contaConsoante`, usando a função `ehVogal`, para contar o número de consoantes. Implemente uma segunda versão usando a função `contaVogal`.

12. Como já sabemos, a função `map` é uma função integrada de Python utilizada para aplicarmos uma função a cada elemento de uma lista, retornando uma nova lista contendo os elementos resultantes da aplicação da função. Por exemplo:

```
>>> import math
>>> lista1 = [1, 4, 9, 16, 25]
>>> lista2 = list(map(math.sqrt, lista1)) #temos que transformar o retorno para
                                          lista usando a função integrada list.
>>> print(lista2)
[1.0, 2.0, 3.0, 4.0, 5.0]
>>> lista3 = list(map(lambda x: x**2, lista2))
```

```
>>> print(lista3)
[1.0, 4.0, 9.0, 16.0, 25.0]
```

Defina uma função, chamada `myMap`, que receba uma função `f` e uma lista e retorna uma nova lista com os elementos modificados pela função `f`. Na implementação da função `myMap`, você **não** pode chamar/usar a função integrada `map`.

13. Outra função integrada é a `filter`. Como o próprio nome já diz, ela filtra os elementos de uma sequência, "deixando passar" para a sequência resultante apenas os elementos para os quais a chamada da função que o usuário passou retornar `True`. Por exemplo,

```
>>> a = [1, 2, 3, 4, 5, 6]
>>> p = list(filter(lambda x : x % 2 == 0, a))
>>> print(p)
[2, 4, 6]
```

Considerando a lista: `l1 = list(range(2, 101))` e usando a função `filter`, crie uma lista `l2` que contenha apenas:

a. Números divisíveis por 3 e 5;

b. Números primos.

14. Usando compreensão de lista, crie uma lista que represente os seguintes conjuntos:

a. $S_1 = \{x^2 + 2 \mid x \in \mathbb{N}, x \leq 10\}$;

b. $S_2 = \{x \mid x \in \mathbb{Z}, -100 \leq x \leq 100, x \text{ é divisível por 3 e 5}\}$;

c. $S_3 = \{x \mid x \in \mathbb{N}, x \leq 500, x \text{ é um número perfeito}\}$. Você deve implementar uma função para verificar se x é perfeito ou não;

d. $S_4 = \{x \mid x \in \mathbb{N}, x \leq 100, x \text{ é um número primo}\}$. Você deve implementar uma função para verificar se um x é primo ou não;

e. $S_5 = \{F_n \mid n \in \mathbb{N}, 1 \leq n \leq 10\}$, onde F_n representa o n-ésimo termo da sequência de Fibonacci.

15. A Regra dos Trapézios é um dos métodos numéricos para calcular o valor de uma integral definida. Ela é dada por:

$$\int_a^b f(x)dx \approx \frac{\Delta x}{2}\left(f(x_0) + 2f(x_1) + 2f(x_2) + \ldots 2f(x_{n-1}) + f(x_n)\right),$$

onde, $\Delta x = \frac{b-a}{n}$ e $x_i = a + i\Delta x$.

Defina uma função, chamada `trapezio`, que implementa a Regra dos Trapézios. Essa função deve ter, no mínimo, os seguintes parâmetros obrigatórios: a função f, o valor de a, o valor de b e o número de subintervalos n. O objetivo da função é retornar uma aproximação de $\int_a^b f(x)dx$.

A função `trapezio` deve, obrigatoriamente, ser uma função de alta ordem (*Higher Order Function*). Veja alguns exemplos de utilização da função:

```
import math as m

def trapezio(<lista de parâmetros>):
  #Implementação

def funcao1(x):
  return x**2 + x + 2

def funcao2(x):
```

```
  return m.exp(x)

def main():
  print(trapezio(funcao1, 0, 10, 10)) #405.0

  print(trapezio(funcao2, -10, 10, 100))  #22099.838400773457

  print(trapezio(lambda x: m.sqrt(x) + m.exp(x), 5, 10, 100)) #21896.23865093272

  print(trapezio(lambda x: m.log(x)**3/x, 1, 15, 100)) #13.445205789652903

  print(trapezio(lambda x: m.sin(x), 0, m.pi, 5))   #1.933765598092805
  print(trapezio(lambda x: m.sin(x), 0, m.pi, 10))  #1.9835235375094546
  print(trapezio(lambda x: m.sin(x), 0, m.pi, 50))  #1.9993419830762615
  print(trapezio(lambda x: m.sin(x), 0, m.pi, 500)) #1.999993420259403
  print(trapezio(lambda x: m.sin(x), 0, m.pi, 900)) #1.999997969216791

  print(trapezio(lambda a: m.sin(a)**2 + 2 * m.sin(2*a)**4, 0, m.pi, 500)) #3.
                                                    926990816987246

main()
```

16. Um dos métodos mais famosos para encontrar raízes (aproximadas) de equações é o Método de Newton-Raphson, dado pela equação abaixo:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x)},$$

sendo x_1 uma aproximação inicial dada. O erro relativo do método é dado por: $\left|\frac{x_{i+1} - x_i}{x_{i+1}}\right|$.

Defina uma função, chamada `newton`, que implementa o Método de Newton-Raphson. Essa função deve ter, no mínimo, os seguintes parâmetros obrigatórios: a função f, a função associada a derivada de f e o valor de uma aproximação inicial da raiz (x_1). O objetivo da função é retornar uma aproximação de uma das raízes de f. O método deve parar (encerar) quando um dos seguinte critérios for atingido:

- $f(x_{i+1}) = 0$;
- O erro relativo for menor ou igual que 10^{-15};
- O número de iterações for maior que 50.

A função `newton` deve, obrigatoriamente, ser uma função de alta ordem (*Higher Order Function*). Veja alguns exemplos de utilização da função:

```
import math as m

def newton(<lista de parâmetros>):
  #Implementação

def F(x):
  return x**2 - 2

def derivadaF(x):
  return 2*x

def main():
  #Encontra uma raiz real do polinomio: x² - 2 = 0
  print(newton(F, derivadaF, 2))
```

```
15    #Encontra uma raiz real do polinomio: x² - 3 = 0
16    print(newton(lambda x: x**2 - 3, lambda x: 2*x, 1))
17    #Encontra uma raiz real da equação: x³ - 3*x² + sin(x) = 0
18    # Raizes: 0, 0.371781, 2.98215
19    #De acordo com o chute inicial, o metodo encontra uma raiz diferente
20    print(newton(lambda x: x**3 - 3*x**2 + m.sin(x), lambda x: 3*x**2 - 6*x + m.
                                                cos(x), -1))
21    print(newton(lambda x: x**3 - 3*x**2 + m.sin(x), lambda x: 3*x**2 - 6*x + m.
                                                cos(x), 1))
22    print(newton(lambda x: x**3 - 3*x**2 + m.sin(x), lambda x: 3*x**2 - 6*x + m.
                                                cos(x), 2))
23    print(newton(lambda x: x**3 - 3*x**2 + m.sin(x), lambda x: 3*x**2 - 6*x + m.
                                                cos(x), 3))
24
25  main()
```