TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 05 (30 minutos) Turno 2ª feira 9:00-10:30
Nome:
Número:
Data:
Curso:

Capítulo 5 - Listas

Escreva uma função chamada `retifica` que recebe uma lista de inteiros `lst`, e devolve a lista obtida da lista original em que todos os inteiros negativos foram substituídos por `0`. Não necessita verificar a validade dos argumentos. Por exemplo,

```
>>> retifica([3, -4, -3, 1, 7])
[3, 0, 0, 1, 7]
```

Solução 1:

```
def retifica(lst):
    res = []
    for i in range(len(lst)):
        if lst[i] < 0:
            res += [0]
        else:
            res += [lst[i]]
    return res
```

Solução 2:

```
def retifica(lst):
   return [e if (e > 0) else 0 for e in lst]
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 05 (30 minutos) Turno 2ª feira 10:30-12:00
Nome:
Número:
Data:
Curso:

Capítulo 5 - Listas

Escreva uma função chamada `valor_absoluto` que recebe uma lista de inteiros `lst`, e devolve a lista obtida da lista original em que todos os inteiros são substituídos pelo seu valor absoluto. Não necessita verificar a validade dos argumentos. Não pode utilizar a função embebida `abs()`. Por exemplo,

```
>>> valor_absoluto([3, -4, -3, 1, 7])
[3, 4, 3, 1, 7]
```

Solução 1:

```
def valor_absoluto(lst):
    res = []
    for i in range(len(lst)):
        if lst[i] < 0:
            res += [-lst[i]]
        else:
            res += [lst[i]]
    return res
```

Solução 2:

```
def valor_absoluto(lst):
   return [e if (e > 0) else -e for e in lst]
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 05 (30 minutos) Turno 5ª feira 10:30-12:00
Nome:
Número:
Data:
Curso:

Capítulo 5 - Listas

Escreva uma função com o nome `remove_repetidos` que recebe uma lista `lst` e devolve a lista obtida da lista original em que todos os elementos repetidos foram removidos. Não necessita verificar a validade dos argumentos. Por exemplo,

```
>>> remove_repetidos([2, 4, 3, 2, 2, 2, 3])
[2, 4, 3]
>>> remove_repetidos([2, 5, 7])
[2, 5, 7]
```

Solução 1 (com for):

```
def remove_repetidos_d(lst):
    # versão destrutiva
    for i in range(len(lst)-1, -1, -1):
        if lst[i] in lst[0:i]:
            del(lst[i])
    return l
```

Solução 2 (sem utilizar o operador del):

```
def remove_repetidos_nd(lst):
    # versão não destrutiva
    newlst = []
    for e in l:
        if e not in newlst:
            newlst += [e]
    return newlst
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 05 (30 minutos) Turno 5ª feira 14:00-15:30
Nome:
Número:
Data:
Curso:

Capítulo 5 - Listas

Escreva uma função chamada `substitui` que recebe uma lista `lst`, e dois valores `velho` e `novo`, e que devolve a lista que resulta de substituir em `lst` todas as ocorrências de `velho` por `novo`. Não necessita verificar a validade dos argumentos. Por exemplo,

```
>>> substitui([1, 2, 3, 2, 4], 2, 'a')
[1, 'a', 3, 'a', 4]
```

Solução 1:

```
def substitui_nd(lst, velho, novo):
    # versão não destrutiva
    res = []
    for e in lst:
        if e == velho:
            res = res + [novo]
        else:
            res = res + [e]
    return res
```

Solução 2:

```
def substitui_d(lst, velho, novo):
    # versão destrutiva
    for i in range(len(lst)):
        if lst[i] == velho:
            lst[i] = novo
    return lst
```

Solução 3:

```
def substitui(lst, velho, novo):
   return [e if e != velho else novo for e in lst]
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 05 (30 minutos) Turno 6ª feira 10:00-11:30
Nome:
Número:
Data:
Curso:

Capítulo 5 - Listas

Escreva uma função chamada `posicoes_lista` que recebe uma lista `lst` e um elemento `elem`, e devolve uma lista contendo todas as posições em que o elemento ocorre na lista. Por exemplo,

```
>>> posicoes_lista(['a', 2, 'b', 'a'], 'a')
[0, 3]
```

Solução 1:

```
def posicoes_lista(lst, elem):
    res = []
    for i in range(len(lst)):
        if lst[i] == elem:
            res = res + [i]
    return res
```

Solução 2:

```
def posicoes_lista(lst, elem):
    return [i for i in range(len(lst)) if lst[i] == elem]
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 05 (30 minutos) Turno 6ª feira 14:00-15:30
Nome:
Número:
Data:
Curso:

Capítulo 5 - Listas

Escreva uma função chamada `centering` que recebe uma lista de inteiros `vetor`, e devolve uma lista contendo os elementos do `vetor` normalizados de forma que o novo vetor tenha média 0, ou seja, cada elemento é atualizado pelo seu valor menos a média do `vetor` de entrada. Não necessita verificar a validade dos argumentos. Por exemplo,

```
>>> centering([1, 2, 3, 4])
[-1.5, -0.5, 0.5, 1.5]
```

Solução 1:

```
def centering_nd(vetor):
    # versao nao destrutiva
    mean = 0
    for elem in vetor:
        mean += elem
    mean = mean/len(vetor)
    res = []
    for elem in vetor :
        res += [ elem - mean ]
    return res
```

Solução 2:

```
def centering_d(vetor):
    # versao destrutiva
    mean = 0
    for elem in vetor:
        mean += elem
    mean = mean/len(vetor)
    for i in range(len(vetor)):
        vetor[i] = vetor[i] - mean
    return vetor
```

Solução 3:

```
def centering_nd2(vetor):
      mean = sum(v)/len(vetor)
      return [e - mean for e in vetor]
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 05 (30 minutos) Turno 6ª feira 15:30-17:00
Nome:
Número:
Data:
Curso:

Capítulo 5 - Listas

Escreva uma função chamada `multiplica_vs` que recebe uma lista de inteiros `vetor` e um inteiro `escalar`, e devolve uma lista contendo o produto de cada elemento do `vetor` pelo `escalar`. Não necessita verificar a validade dos argumentos. Por exemplo,

```
>>> multiplica_vs([3, 4, 0, 1, 7], 2)
[6, 8, 0, 2, 14]
```

Solução 1:

```
def multiplica_vs(vetor, escalar):
    res = []
    for e in vetor:
        res += [ e*escalar ]
    return res
```

Solução 2:

```
def multiplica_vs(vetor, escalar):
    return [e*escalar for e in vetor]
```