TÉCNICO LISBOA

Fundamentos de Programação - 2019/2020 Aula Prática 05 (30 minutos) Turno 6ª feira 15:30-17:00
Nome:
Número:
Data:
Curso:

Capítulo 5 - Listas

Escreva uma função em Python com o nome `duplica_elementos` que recebe uma lista e devolve a lista obtida da lista original em que todos os elementos são duplicados. Não necessita verificar a validade dos argumentos. Por exemplo,

```
>>> duplica_elementos(['a', ['b', 'c'], 5])
['a', 'a', ['b', 'c'], ['b', 'c'], 5, 5]
>>> duplica_elementos([])
[]
```

Solução 1:

```
def duplica_elementos(lst):
    res = []
    for e in lst:
        res = res + [e, e]
    return res
```

Solução 2:

```
def duplica_elementos(lst):
   return [e for e in lst for n in range(2)]
```

TÉCNICO LISBOA

Fundamentos de Programação - 2019/2020 Aula Prática 05 (30 minutos) Turno 6ª feira 14:00-15:30
Nome:
Número:
Data:
Curso:

Capítulo 5 - Listas

Escreva uma função chamada `crescimento` que recebe uma lista de inteiros `lst`, e devolve a lista obtida da lista original em que todos os inteiros são multiplicados pela posição que ocupam na lista. Não necessita verificar a validade dos argumentos. Por exemplo,

```
>>> crescimento([3, -4, -3, 1, 7])
[0, -4, -6, 3, 28]
```

Solução 1:

```
def crescimento(lst):
    res = []
    for i in range(len(lst)):
        res += [i*lst[i]]
    return res
```

Solução 2:

```
def crescimento(lst):
    return [lst[i]*i for i in range(len(lst))]
```

TÉCNICO LISBOA

Fundamentos de Programação - 2019/2020 Aula Prática 05 (30 minutos) Turno 2ª feira 09:00-10:30	
Nome:	
Número:	
Data:	
Curso:	

Capítulo 5 - Listas

Escreva uma função chamada `dups_vetor` que recebe uma lista de inteiros `lst`, e devolve uma lista em que os valores iguais consecutivos foram eliminados. Não necessita verificar a validade dos argumentos. Por exemplo,

```
>>> dups_vetor([5,2,2,2,4,6,6,2,2,8])
[5, 2, 4, 6, 2, 8]
```

Solução:

```
def dups_vetor_d(lst):
  # versão destrutiva
  for x in range(len(lst)-1,0,-1) :
    if lst[x-1] == lst[x] :
      del(lst[x])
  return lst
```

Solução:

```
def dups_lst_nd(lst):
      # versão não destrutiva
      newlst = []
      for e in lst:
      if not newlst or e != newlst[-1]:
            newlst += [e]
      return newlst
```

TÉCNICO LISBOA

Fundamentos de Programação - 2019/2020 Aula Prática 05 (30 minutos) Turno 2ª feira 10:30-12:00	
Nome:	
Número:	
Data:	
Curso:	

Capítulo 5 - Listas

Escreva uma função chamada `inverte` que recebe uma lista de inteiros `lst`, e devolve uma lista contendo os elementos pela ordem inversa. Não pode usar as funções `reverse()` ou `reversed()`, nem o operador *slice*. Não necessita verificar a validade dos argumentos. Por exemplo,

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

Solução:

```
def inverte_d(lst):
  # versão destrutiva
  for x in range(len(lst)//2) :
    lst[x], lst[-x-1] = lst[-x-1], lst[x]
  return lst
```

Solução:

```
def inverte_nd(lst):
  # versão não destrutiva
  newlst = []
  for e in lst:
    newlst = [e] + newlst
  return newlst
```

TÉCNICO LISBOA

Fundamentos de Programação - 2019/2020 Aula Prática 05 (30 minutos) Turno 4ª feira 08:00-09:30	
Nome:	
Número:	
Data:	
Curso:	

Capítulo 5 - Listas

Escreva uma função chamada `mult_vetores` que recebe duas listas de inteiros (`a` e `b`), e devolve uma lista onde cada elemento resulta da multiplicação dos elementos de `a` e `b` na mesma posição. Se as listas não tiverem o mesmo comprimento, deve ser devolvida a lista vazia. Não necessita verificar a validade dos argumentos. Por exemplo,

```
>>> mult_vetores([3, 4, 0, 1, 7], [2, 3])
[]
>>> mult_vetores([3, 4, 0, 1, 7], [2, 3, 6, 8, 2])
[6, 12, 0, 8, 14]
```

Solução:

```
def mult_vetores(a, b) :
  if len(a) != len(b) :
    return []
  for x in range(len(b)) :
    a[x] *= b[x]
  return a
```

TÉCNICO LISBOA

Fundamentos de Programação - 2019/2020 Aula Prática 05 (30 minutos) Turno 5ª feira 10:30-12:00	
Nome:	
Número:	
Data:	
Curso:	

Capítulo 5 - Listas

Escreva uma função chamada `vpow` que recebe uma lista de inteiros (`vetor`) e um inteiro (`escalar`), e devolve uma lista onde cada elemento é a potência do `vetor` pelo `escalar`. Não necessita verificar a validade dos argumentos. Por exemplo,

```
>>> vpow([1,2,3,4], 3)
[1, 8, 27, 64]
```

Solução 1:

```
def vpow_nd(vetor, escalar):
    # versao nao destrutiva
    res=[]
    for elem in vetor:
        res += [ elem ** escalar ]
    return res
```

Solução 2:

```
def vpow_d(vetor, escalar):
    # versao destrutiva
    res=[]
    for i in range(len(vetor)):
        vetor[i] = vetor[i] ** escalar
    return vetor
```

TÉCNICO LISBOA

Fundamentos de Programação - 2019/2020 Aula Prática 05 (30 minutos) Turno 6ª feira 12:30-14:00
Nome:
Número:
Data:
Curso:

Capítulo 5 - Listas

Escreva uma função chamada `norm` que recebe uma lista de inteiros (`vetor`), e devolve uma lista contendo os elementos do `vetor` escalados no intervalo `[0, 1]`, ou seja `(elem-low)/(high-low)`. Não necessita verificar a validade dos argumentos. Por exemplo,

```
>>> norm([1,2,3,4,5])
[0.0, 0.25, 0.5, 0.75, 1.0]
```

Solução 1:

```
def norm_nd(vetor):
    # versao nao destrutiva
    low = min(vetor)
    high = max(vetor)
    res = []
    for elem in vetor :
      res += [ (elem-low) / (high-low) ]
    return res
```

Solução 1:

```
def norm_d(vetor):
    # versao destrutiva
    low = min(vetor)
    high = max(vetor)
    for i in range(len(vetor)):
       vetor[i] = (vetor[i]-low) / (high-low)
    return vetor
```