TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 06 (30 minutos) Turno 2ª feira 09:00-10:30
Nome:
Número:
Data:
Curso:

Capítulo 8 - Dicionários

Escreva a função `separa` que recebe uma lista com números inteiros e um inteiro, e que devolve um dicionario com a chave 'menores' associada à lista contendo os números menores ou iguais que o valor inteiro passado como argumento, e a chave 'maiores' associada à lista contendo os números maiores que o argumento. Não necessita validar os argumentos. Por exemplo,

```
>>> lista = [1,2,3,4,5,6,7,8,9]
>>> separa(lista, 3)
{'menores': [1, 2, 3], 'maiores': [4, 5, 6, 7, 8, 9]}
```

Solução:

```
def separa(lista, valor):
    dic = {'menores' : [], 'maiores' : []}
    for e in lista:
        if e <= valor:
            dic['menores'] += [e]
        else:
            dic['maiores'] += [e]
    return dic

def separa_lc(lista, valor):
   return {'menores': [ e for e in lista if e <= valor ],
           'maiores': [ e for e in lista if e > valor ]}
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 06 (30 minutos) Turno 2ª feira 10:30-12:00
Nome:
Número:
Data:
Curso:

Capítulo 8 - Dicionários

Escreva a função `min_dict` que recebe um dicionário (cujos valores associados às chaves são listas não vazias contendo números inteiros) e devolve um dicionário com o mínimo dos valores associados a cada chave. Não necessita validar os argumentos. Por exemplo,

```
>>> d1 = {'a' : [1, 2, 3, 4], 'b' : [3, 4]}
>>> min_dict(d1)
{'a': 1, 'b': 3 }
```

Solução:

```
def min_dict(d):
    nd = {}
    for c in d:
        min = d[c][0]
        for e in d[c]:
            if e < min:
                min = e
        nd[c] = min
    return nd
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 06 (30 minutos) Turno 5ª feira 10:30-12:00
Nome:
Número:
Data:
Curso:

Capítulo 8 - Dicionários

Escreva uma função em Python que recebe um dicionário cujos valores associados às chaves correspondem a listas de inteiros e que devolve o dicionário que se obtém "invertendo" o dicionário recebido, no qual as chaves são os inteiros que correspondem aos valores do dicionário original e os valores são as chaves do dicionário original às quais os valores estão associados. Não necessita validar os argumentos. Por exemplo:

```
>>> inverte_dic({'a': [1, 2], 'b': [1, 5], 'c': [9], 'd': [4]})
{1: ['a', 'b'], 2: ['a'], 4: ['d'], 5: ['b'], 9: ['c']}
```

Solução:

```
def inverte_dic(d):
  res = {}
  for e in d:
    for v in d[e]:
      if v in res:
        res[v] = res[v] + [e]
      else:
        res[v] = [e]
  return res
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 06 (30 minutos) Turno 5ª feira 14:00-15:30
Nome:
Número:
Data:
Curso:

Capítulo 8 - Dicionários

Escreva a função `chave_max` que recebe um dicionário (cujos valores associados às chaves são listas não vazias contendo números inteiros positivos ou negativos), e devolve um tuplo contendo a chave ou chaves que contém o valor máximo. Não necessita validar os argumentos. Por exemplo,

```
>>> d = {'a' : [1, -2, 3], 'b' : [3, -1]}
>>> chave_max(d)
('a', 'b')
>>> d = {'a' : [1, -2, 3], 'b' : [4, -1]}
>>> chave_max(d)
('b')
```

Solução:

```
def chave_max(d):

      k = d.keys()[0]
      max = d[k][0]
      for k in d:
            for e in d[k]:
                  if e > max:
                        max = e
      t = ()
      for k in d:
            if max in d[k]:
                  t += (k,)

      return t
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 06 (30 minutos) Turno 6ª feira 10:00-11:30
Nome:
Número:
Data:
Curso:

Capítulo 8 - Dicionários

Escreva a função `junta` que recebe dois dicionários, cujos valores associados às chaves correspondem a listas, e devolve o dicionário que contém todas as chaves contidas em pelo menos um dos dicionários e o valor associado a cada chave corresponde à lista obtida pela "união" (no sentido de conjuntos) das listas correspondendo às chaves existentes nos dicionários. Não necessita validar os argumentos. Por exemplo,

```
>>> d1 = {'a' : [1, 2], 'b' : [3, 4]}
>>> d2 = {'b' : [4, 5], 'c' : [6, 7]}
>>> junta(d1, d2)
{'a': [1, 2], 'b': [3, 4, 5], 'c': [6, 7]}
```

Solução:

```
def junta(d1, d2):
    for c in d2:
       if c in d1:
            for el in d2[c]:
               if el not in d1[c]:
                    d1[c] = d1[c] + [el]
       else:
            d1[c] = d2[c]
     return d1
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 06 (30 minutos) Turno 6ª feira 14:00-15:30
Nome:
Número:
Data:
Curso:

Capítulo 8 - Dicionários

Escreva a função `maior_valor` que recebe um dicionário, cujos valores associados às chaves podem ser inteiros positivos ou listas (não vazias) de inteiros positivos, e que devolve o maior inteiro existente como valor no dicionário. Não necessita validar os argumentos. Por exemplo,

```
>>> maior_valor({'a' : 3, 'b' : [6, 3, 90], 'c' : 20, \
                 'd' : [1, 33, 12]})
90
```

Solução 1:

```
def maior_valor(d):
    maior = 0
    for k in d: # calcula-se o maior
        if isinstance(d[k], int): # it's an int
            if d[k] > maior:
                maior = d[k]
        else:                     # it's a list of int
            for e in d[k]:
                if e > maior:
                    maior = e
     return maior
```

Solução 2:

```
def maior_valor(d):
    lvals = []
    for k in d: # calcula-se o maior
        if isinstance(d[k], int): # it's an int
            lvals += [ d[k] ]
        else:                     # it's a list of int
            for e in d[k]:
                lvals += [ e ]
     return max(lvals)
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 06 (30 minutos) Turno 6ª feira 15:30-17:00
Nome:
Número:
Data:
Curso:

Capítulo 8 - Dicionários

Escreva a função `soma_valores` que recebe um dicionário cujos valores associados às chaves são ou inteiros ou listas (não vazias) de inteiros e que devolve a soma de todos os valores existentes no dicionário.Não necessita validar os argumentos. Por exemplo,

```
>>> soma_valores({'a' : 3, 'b' : [6, 3, 90], 'c' : 20, \
                  'd' : [0, 33, 12]})
167
```

Solução:

```
def soma_valores(d):
    soma = 0
    for c in d:
        if isinstance(d[c], int):
            soma = soma + d[c]
        else:
            for e in d[c]:
                soma = soma + e
    return soma
```