TÉCNICO LISBOA

Fundamentos de Programação - 2019/2020 Aula Prática 06 (30 minutos) Turno 6ª feira 14:00-15:30
Nome:
Número:
Data:
Curso:

Capítulo 8 - Dicionários

Escreva a função `filter_by_elem` que recebe um dicionário (cujos valores associados às chaves são listas não vazias contendo números inteiros positivos) e um elemento, e devolve o subconjunto do dicionário cujas listas associadas não contenham o elemento. Não necessita validar os argumentos. Por exemplo,

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

Solução:

```
def filter_by_elem(d, e):
    newd = {}
    for c in d:
        if e not in d[c]:
            newd[c] = d[c]
    return newd
```

TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 Aula Prática 06 (30 minutos) Turno 2ª feira 10:30-12:00
Nome:
Número:
Data:
Curso:

Capítulo 8 - Dicionários

Escreva a função `media_chave` que recebe um dicionário (cujos valores associados às chaves são listas não vazias contendo números inteiros) e devolve um dicionário com a média dos valores associados a essa chave. Por exemplo,

```
>>> d1 = {'a' : [1, 2, 3, 4], 'b' : [3, 4]}
>>> media_chave(d1)
{'a': 2.5, 'b': 3.5 }
```

Solução:

```
def media_chave(d):
    for c in d:
        total = 0
        for e in d[c]:
            total += e
        d[c] = total/len(d[c])
    return d
```

TÉCNICO LISBOA

Fundamentos de Programação - 2019/2020 Aula Prática 06 (30 minutos) Turno 2ª feira 09:00-10:30
Nome:
Número:
Data:
Curso:

Capítulo 8 - Dicionários

Escreva a função `chave_lista_maior` que recebe um dicionário não vazio, cujos valores associados às chaves são listas, e devolve a chave da lista que possui maior comprimento. Em caso de dois comprimentos iguais retorna a última chave. Não necessita validar os argumentos. Por exemplo,

```
>>> d1 = {'a' : [1, 2, 3, 4], 'b' : [3, 1], 'c' : [1, 5,
'batata']}
>>> chave_lista_maior(d1)
a
```

Solução:

```
def chave_lista_maior(d):
    maior = 0
    for c in d:
        if len(d[c]) >= maior:
            maior = len(d[c])
            chave = c
    return chave
```

TÉCNICO LISBOA

Fundamentos de Programação - 2019/2020 Aula Prática 06 (30 minutos) Turno 4ª feira 08:00-09:30
Nome:
Número:
Data:
Curso:

Capítulo 8 - Dicionários

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

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

Solução:

```
def maximo_chave(d, c):
    maximo = d[c][0]
    for n in d[c]:
        if n > maximo:
            maximo = n
    return maximo
```

TÉCNICO LISBOA

Fundamentos de Programação - 2019/2020 Aula Prática 06 (30 minutos) Turno 5ª feira 10:30-12:00
Nome:
Número:
Data:
Curso:

Capítulo 8 - Dicionários

Escreva a função `separa_pares_impares` que recebe uma lista, com inteiros positivos, e que devolve um dicionario com as chaves 'pares' e 'impares' onde guarda os numeros pares e impares, respetivamente. Não necessita validar os argumentos. Por exemplo,

```
>>> lista = [1,2,3,4,5,6,7,8,9]
>>> separa_pares_impares(lista)
{'pares': [2, 4, 6, 8], 'impares': [1, 3, 5, 7, 9]}
```

Solução:

```
def separa_pares_impares(lista):
    dic = {'pares' : [], 'impares' : []}
    for e in lista:
        if e%2==0:
            dic['pares'] += [e]
        else:
            dic['impares'] += [e]
    return dic
```

TÉCNICO LISBOA

Fundamentos de Programação - 2019/2020 Aula Prática 06 (30 minutos) Turno 6ª feira 12:30-14:00
Nome:
Número:
Data:
Curso:

Capítulo 8 - Dicionários

Escreva a função `conta_ocorrencias` que recebe uma lista, com inteiros, e que devolve um dicionario com as chaves que corresponde a inteiro e o valor ao número de ocorrências desse inteiro na lista. Não necessita validar os argumentos. Por exemplo,

```
>>> lista = [1,1,2,1,20]
>>> conta_ocorrencias(lista)
{1: 3, 2: 1, 20: 1}
```

Solução:

```
def conta_ocorrencias(lista):
      dic = {}
      for e in lista:
            if e not in dic:
                  dic[e] = 1
            else:
                  dic[e] += 1
      return dic
```