MBA em Ciência de Dados

Programação para Ciência de Dados

Avaliação Python Parte II

Material Produzido por Luis Gustavo Nonato
Cemeai - ICMC/USP São Carlos

As respostas devem ser fornecidas no Moodle. O notebook é apenas para a implementação dos códigos que fornecerão as respostas

Exercício 1)

Considere a lista `mat` abaixo, onde cada elemento é também uma lista. A lista correspondendo a cada elemento de `mat` tem como primeiro elemento ao cargo, o segundo elemento o nível de formação e o terceiro o nome de um funcionário.

Utilize comprehension para gerar uma lista chamada `nomes` que contenha o nome dos funcionários que **NÃO** são doutor. O código correto é para gerar a lista `nomes` é:

a) nomes = [s[2] for s in mat if s[1] != 'doutor']
b) nomes = [s[1] for s in mat if s[2] != 'doutor']
c) nomes = [s for s in mat if s != 'doutor']
c) nomes = [s[0] for s in mat if s != 'doutor']

In [1]:
```
mat = [
    ['supervisor', 'tecnico', 'Carlos'],
    ['assistente', 'tecnico', 'Lucas'],
    ['iniciante', 'doutor', 'Jeremias'],
    ['supervisor', 'mestre', 'Alberto'],
    ['gerente', 'graduado', 'Ricardo'],
    ['engenheiro', 'graduado', 'Fernando'],
]

nomes = [s[2] for s in mat if s[1] != 'doutor' ]
print(nomes)
```

```
['Carlos', 'Lucas', 'Alberto', 'Ricardo', 'Fernando']
```

Exercício 2)

Ainda utilizando a lista `mat` do exercício anterior e comprehension, crie um dicionário da forma `nome:(cargo, area)` , ou seja, as chaves são os *nomes* e os valores uma tupla com o par (*cargo*, *nível de formação*). No dicionário produzido, o valor associado à chave 'Carlos' será:

a) ('supervisor', 'tecnico')
b) ('tecnico', 'assistente')
c) ('supervisor', 'mestre')
d) ('mestre','supervisor')

In [5]:
```
dc = {s[2]:(s[0],s[1]) for s in mat}
print('Carlos:',dc['Carlos'])
print(dc)
```

```
Carlos: ('supervisor', 'tecnico')
{'Carlos': ('supervisor', 'tecnico'), 'Lucas': ('assistente', 'tecnico'), 'Jer
emias': ('iniciante', 'doutor'), 'Alberto': ('supervisor', 'mestre'), 'Ricardo
': ('gerente', 'graduado'), 'Fernando': ('engenheiro', 'graduado')}
```

Exercício 3)

A função `variacoes` descrita abaixo

```
def variacoes(s):
    dc = {________________}
    return(dc)
```

recebe uma 'string' como parâmetro e gera um dicionário como resposta, onde as chaves do dicionário são todas as variações da string com um caracter removido, sendo o valor associado a cada chave a string original. Por exemplo:

```
print(variacoes('casa'))
```

deve resultar em:

```
{'asa': 'casa', 'csa': 'casa', 'caa': 'casa', 'cas': 'casa'}
```

Qual das alternativas abaixo completa corretamente a função?

a) {s[0]+s[1] for s in range(len(s))}
b) {s[:i]+s[i+1:]:i for i in s}
c) {s[0:i]+s[i+1:]:s for i in len(s)}
d) {s[0:i]+s[i+1:]:s for i in range(len(s))}

In [18]:
```
def variacoes(s):
    dc = {s[0:i]+s[i+1:]:s for i in range(len(s))}
    return(dc)

print(variacoes('casamento'))
```

```
{'asamento': 'casamento', 'csamento': 'casamento', 'caamento': 'casamento', 'c
asmento': 'casamento', 'casaento': 'casamento', 'casamnto': 'casamento', 'casa
meto': 'casamento', 'casameno': 'casamento', 'casament': 'casamento'}
```

Exercício 4)

O método sorted assume como parâmetos uma sequência e um parâmetro `key` que pode ser uma função, a qual é aplicada aos elementos da lista a fim de ordená-los de acordo com os valores resutantes da função. Utilize uma função lambda como parâmetro para método sorted para ordenar os elementos de uma lista como se estes fossem da forma $(x + 0.5)^2$.

O resultado da função construída quando aplicada à lista

```
[0.11, -0.11, 0.4, 0.11, -0.57, -0.05, 0.85, -0.27, -0.07, -0.78]
```

será:

a) [-0.78, -0.57, -0.27, -0.11, -0.07, -0.05, 0.11, 0.11, 0.4, 0.85]
b) [-0.57, -0.27, -0.78, -0.11, -0.07, -0.05, 0.11, 0.11, 0.4, 0.85]
c) [-0.05, -0.07, 0.11, -0.11, 0.11, -0.27, 0.4, -0.57, -0.78, 0.85]
d) [-0.05, 0.11, -0.07, -0.11, 0.11, -0.27, 0.4, -0.57, -0.78, 0.85]

In [5]:
```
l = [0.11, -0.11, 0.4, 0.11, -0.57, -0.05, 0.85, -0.27, -0.07, -0.78]
print(sorted(l,key=lambda x:(x+0.5)**2))
```

```
[-0.57, -0.27, -0.78, -0.11, -0.07, -0.05, 0.11, 0.11, 0.4, 0.85]
```

Exercício 5)

Construa uma função `concatena_dicionarios` que recebe dois dicionários como parâmetros e concatena ambos em um único dicionário contendo as chaves e valores dos dois dicionários. Se a chave se repete em ambos, junte os valores em uma lista. Por exemplo, dados os dicionários:

```
dc1 = {1: '5479', 5: '1479', 7: '1549', 9: '1547'}
dc2 = {5: '2647', 2: '9647', 1: '9247'}
```

a função deve resultar no dicionário

```
{1: ['5479', '9247'], 7: '1549', 5: ['1479', '2647'], 9: '1547', 2: '9647'}
```

Supondo os dicionários:

```
dc1 = {4: '10153', 18: '41118', 2: '101515', 15: '354', 7: '4145', 8: '10316', 3: '
121016',
       11: '15122', 12: '16158', 10: '4188', 14: '15318', 5: '11711', 16: '181412'}

dc2 = {19: '16619', 4: '11310', 18: '171112', 11: '171817', 12: '171911', 6: '4195
',
       5: '171910', 3: '10310', 16: '191916', 10: '111619', 17: '51016'}
```

como entrada para a função construída, quais serão os valores associados as chaves 3 e 18?

a) valor da chave 3 = ['16158', '171911']; valor da chave 18 = ['15122', '171817']
b) valor da chave 3 = ['10153', '11310']; valor da chave 18 = '41118'
c) valor da chave 3 = ['121016', '10310']; valor da chave 18 = ['41118', '171112']
d) valor da chave 3 = ['4188', '111619']; valor da chave 18 = ['181412', '191916']

In [4]:
```
# import numpy as np
# n1 = 25
# c1 = np.random.randint(0,20,n1)
# dc1 = {c1[i]:''.join(str(c) for c in np.random.choice(c1,size=3)) for i in ra
nge(n1)}
# print(dc1)

# n2 = 20
# c2 = np.random.randint(0,20,n2)
# dc2 = {c2[i]:''.join(str(c) for c in np.random.choice(c2,size=3)) for i in ra
nge(n2)}
# print(dc2)

def concatena_dicionarios(d1,d2):
    dr = d1
    for k,v in d2.items():
        if k in d1:
            dr[k] = [dr[k],v]
        else:
            dr[k] = v
    return(dr)

dc1 = {4: '10153', 18: '41118', 2: '101515', 15: '354', 7: '4145', 8: '10316',
3: '121016',
 11: '15122', 12: '16158', 10: '4188', 14: '15318', 5: '11711', 16: '181412'}
dc2 = {19: '16619', 4: '11310', 18: '171112', 11: '171817', 12: '171911', 6: '4
195',
       5: '171910', 3: '10310', 16: '191916', 10: '111619', 17: '51016'}

dc12 = concatena_dicionarios(dc1,dc2)
print(dc12[3],dc12[18])
print(sorted(dc12.items()))
```

```
['121016', '10310'] ['41118', '171112']
[(2, '101515'), (3, ['121016', '10310']), (4, ['10153', '11310']), (5, ['11711
', '171910']), (6, '4195'), (7, '4145'), (8, '10316'), (10, ['4188', '111619
']), (11, ['15122', '171817']), (12, ['16158', '171911']), (14, '15318'), (15,
'354'), (16, ['181412', '191916']), (17, '51016'), (18, ['41118', '171112']),
(19, '16619')]
```