MC346 - Paradigmas de Programação

2016-06-15

Professor: João Meidanis

139546 - João Lopes

Prova 3 (Python) - Resolução

Questão 1

(Valor 2,5) Escreva uma função **reps** em Python que recebe uma lista e retorna uma outra lista contendo apenas os elementos que aparecem duas ou mais vezes na lista de entrada. Exemplos:

```
>>> reps([1,4,2,3,4,2,3,4])
[2,3,4]
>>> reps([1,2,3,4,5])
[]
```

Resposta

```
def reps(lista):
    conjunto = set() # guarda cada elemento
    resultado = set() # assegura unicidade da saída

    for elemento in lista:
        if elemento in conjunto:
            resultado.add(elemento)
        else:
            conjunto.add(elemento)

    return list(resultado)
```

Questão 2

(Valor 2,5) Escreva uma função **brancos** em Python que recebe o nome de um arquivo texto e retorna uma das seguintes 3 coisas:

1. a *string* **"falha ao abrir"** se ocorrer algum problema na abertura do arquivo
2. a *string* **"falha ao ler"** se ocorrer algum problema na leitura do arquivo
3. o número total de caracteres iguais a branco " " que há no arquivo com o nome dado.

Seu programa deve assegurar-se de que o arquivo esteja fechado ao final da rotina, em todos os casos acima.

Resposta

```
def brancos(filename):
    try:
        f = open(filename, ('r')) # read-only
    except Exception:
        return "falha ao abrir"

    try:
        fileString = f.read()
    except Exception:
        f.close()
        return "falha ao ler"

    count = fileString.count(' ', 0, len(fileString))
    f.close()

    return count
```

Questão 3

(Valor 2,5) Escreva uma função **inva** em Python que recebe um dicionário **d** e retorna um dicionário "inverso" do dicionário dado, onde, a cada valor **v** de **d** está associada a lista das chaves de **d** que levam a **v**. Exemplos:

```
>>> inva( {1:2, 3:1, 4:2} )
{2: [1, 4], 1: [3]}
>>> inva( {} )
{}
>>> inva( {2:1, 1:2} )
{1: [2], 2: [1]}
```

Resposta

```
def inva(dic):
    rdic = {} # reversed dictionary
    for k, v in dic.items():
        if rdic.has_key(v):
            rdic.get(v,[]).append(k)
        else:
            rdic.update({v:[k]})

    return rdic
```

Questão 4

(Valor 2,5) Escreva uma classe **Intervalo** que representa um intervalo de coordenadas inteiras e identificador também inteiro. Os objetos desta classe devem ser criados passando-se três parâmetros nominais **id**, **xmin** e **xmax** com valores inteiros. A classe deve possuir também um método elo que recebe um outro intervalo desta classe e calcula o valor do elo entre os dois,

definido como segue:

1. se os intervalos têm interseção não vazia, o elo é o tamanho desta interseção;
2. se os intervalos **a** e **b** não se encontram, o elo é o negativo da distância entre eles, definida como a menor distância entre um ponto de **a** e um ponto de **b**.

Exemplos:

```
>>> a = Intervalo(id=1, xmin=2, xmax=3)
>>> b = Intervalo(id=2, xmin=3, xmax=4)
>>> a.elo(b)
0
>>> b.elo(Intervalo(id=4, xmin=8, xmax=15))
-4
>>> a.elo(Intervalo(id=8, xmin=2, xmax=4))
1
```

Resposta

```
class Intervalo:
    __ID = None
    __begin = None
    __end = None

    def get_id(self):
        return self.__ID
    def get_xmin(self):
        return self.__begin
    def get_xmax(self):
        return self.__end

    def __init__(self, id, xmin, xmax):
        self.__ID = id
        self.__begin = min(xmin, xmax)
        self.__end = max(xmin, xmax)

    def elo(self, intervalo):
        return min(self.__end - intervalo.get_xmin(), intervalo.get_xmax() -
self.__begin)
```