Aula prática nº 09 – Conjuntos, definições por compreensão e geradores

Exercícios

1. Usando o Python em modo interativo, execute as instruções abaixo e interprete os resultados. Tente prever os resultados de cada expressão

```
Lx = [1, 3, 5, 7, 9]
[10+x for x in Lx]
Ly = [2, 4, 6]
[x+y for x in Lx for y in Ly]
{x+y for x in Lx for y in Ly}
[(x,y) for x in Lx for y in Ly]
[(x,y) for y in Ly for x in Lx]
[x*c for c in "abc" for x in Lx]
```

```
[x%3==0 for x in Lx]
[(x,x//3) for x in Lx if x%3==0]
{x:x//3 for x in Lx if x%3==0}
[(x,y) for x in Lx
       for y in Ly if x<y]
{ x:[y for y in Ly if x<y]
       for x in Lx }
any(x%2==0 for x in Lx)
```

2. O programa `imctable2.py` define uma lista com informação dos nomes, pesos e alturas de diversas pessoas e usa uma *list comprehension* para obter uma lista com os nomes apenas. Substitua as reticências por outras *list comprehensions* que produzam:

   a) Uma lista com os valores de IMC de todas as pessoas.

   b) Uma lista de tuplos das pessoas com altura superior a 1.7m.

   c) Uma lista com os nomes das pessoas com IMC entre 18 e 25.

3. O ficheiro `names.txt` tem uma lista de nomes completos de pessoas, com um nome por linha. Escreva um programa que mostre, para cada apelido (último nome), o conjunto de primeiros nomes encontrados na lista, sem repetições. O excerto abaixo é um exemplo do resultado pretendido. *Sugestão: construa um dicionário com chave = último nome e vá acrescentando os primeiros nomes ao conjunto associado a cada chave. Este é um problema que não se consegue reduzir facilmente a uma definição por compreensão.*

```
INACIO : {'ROMEU'}
SA : {'JOAO'}
AMARAL : {'SOLANGE', 'RICARDO'}
MONTEIRO : {'RICARDO', 'PAULO', 'BRUNO'}
```

4. Crie uma função `primesUpTo(n)` que devolva um conjunto com todos os números primos até `n`. Use o algoritmo do *crivo de Eratóstenes*: comece com o conjunto {2, 3, …, n}, depois elimine os múltiplos de 2 a começar em 2^2, depois os múltiplos de 3 a começar em 3^2, pode saltar o 4 porque já foi eliminado (bem como todos os seus múltiplos), depois continue eliminando os múltiplos de cada número que ainda se mantenha no conjunto. No fim, o conjunto conterá apenas os primos. (*Este algoritmo também pode ser implementado sobre uma lista de valores Booleanos, que é uma forma alternativa de representar conjuntos*.)

5. O programa `interests.py` tem uma tabela (dicionário) com os interesses de um conjunto de pessoas. Substitua as reticências por expressões adequadas para:

   a) Criar um dicionário com os interesses comuns a cada par de pessoas. Ou seja, para cada par de pessoas, deve associar o conjunto dos interesses comuns a ambos. Note que se incluir o par (X, Y) não deve incluir (Y, X).

   b) Achar o maior número de interesses em comum. *Sugestão: use a função `max` e uma expressão geradora que percorra o dicionário criado na alínea anterior*.

   c) Criar uma lista dos pares de pessoas que têm o número máximo de interesses comuns.

   d) Criar uma lista de pares de pessoas com menos de 25% de similaridade de interesses. Para medir a similaridade, use o *índice de Jaccard* entre dois conjuntos, que é dado pela razão entre o tamanho da interseção e o tamanho da união entre os conjuntos. O resultado esperado é o seguinte.

```
a) Table of common interests:
{('Paolo', 'Teresa'): {'music', 'writing'}, ('Frank', 'Maria'): {'writing',
'running'}, ('Marco', 'Teresa'): {'writing', 'music'}, ('Frank', 'Teresa'):
{'writing', 'music'}, ('Anna', 'Paolo'): set(), ('Maria', 'Teresa'): {'writing'},
('Anna', 'Frank'): {'reading', 'running'}, ('Frank', 'Paolo'): {'eating', 'music',
'writing'}, ('Anna', 'Marco'): {'reading', 'running'}, ('Frank', 'Marco'):
{'reading', 'writing', 'running', 'music'}, ('Marco', 'Maria'): {'writing',
'running'}, ('Anna', 'Maria'): {'running', 'movies'}, ('Marco', 'Paolo'): {'music',
'writing'}, ('Maria', 'Paolo'): {'writing'}, ('Anna', 'Teresa'): set()}
b) Maximum number of common interests:
4
c) Pairs with maximum number of matching interests:
```