Trabalho de casa 4

Erasmus, alguém?

Programação II

Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática
Licenciatura em Tecnologias da Informação

Entrega a 25 de maio de 2017

Certamente que já ouviram falar de Erasmus. Não o Erasmo de Roterdão, mas o programa de intercâmbio universitário, agora chamado Erasmus+. Este programa está aberto a estudantes, docentes e outros funcionários do ensino superior. Há tanta coisa que gostariamos de saber sobre este programa: Que país envia mais estudantes? E que país recebe mais estudantes? Quais as línguas mais utilizadas nos programas de intercâmbio? Estas e muitas outras perguntas estão respondidas em brochuras da Comissão Europeia. Por exemplo, espreitem *A Statistical Overview of the Erasmus Programme in 2012–13*. Este trabalho pretende reproduzir (e melhorar!) alguns dos gráficos constantes neste tipo de brochura.

O ponto de partida é a informação disponibilizada pela Comissão Europeia. Entramos em *European Union European Union Open Data Portal* e procuramos *erasmus*. Se tudo correr bem a primeira meia dúzia de resultados aponta para estatísticas sobre os últimos anos letivos. Por exemplo, a página *Erasmus mobility statistics 2012-13* permite descarregar um ficheiro CSV com os dados sobre mobilidade de estudantes: *Raw data of Erasmus student mobility (study exchanges and work placements in 2012–13)*.

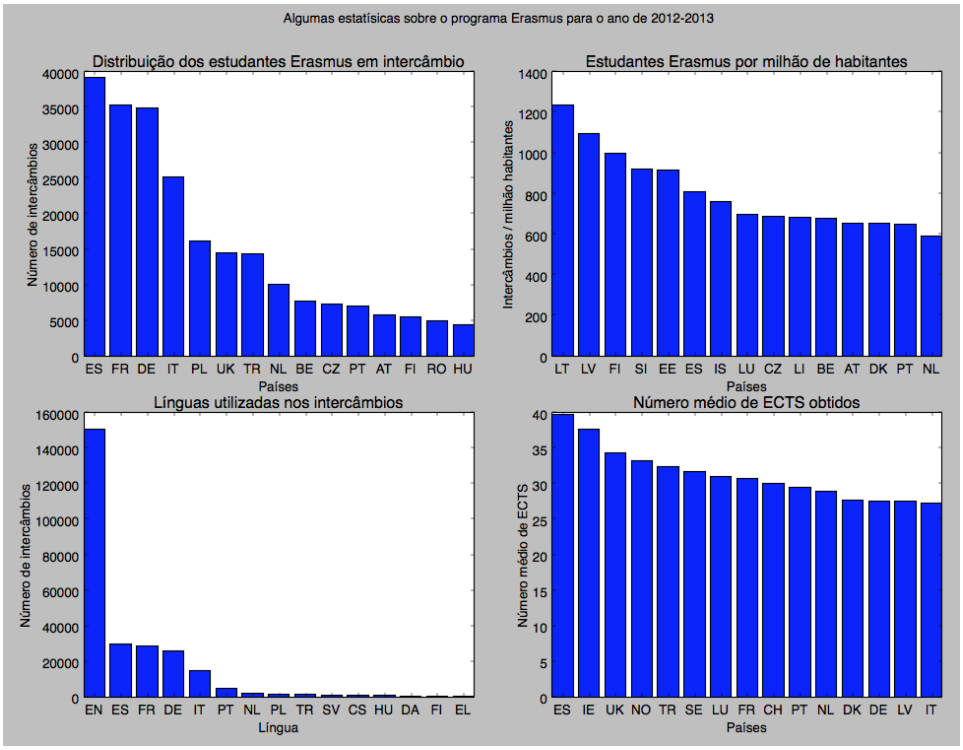
Vamos também precisar de informação referente aos vários países do planeta. O sítio *WorldData.info* disponibiliza dados sobre uma série de

indicadores. Em particular, em *CVS downloads* encontram um ficheiro que contém a população de cada país.

Baseado num ficheiro de informação sobre o programa Erasmus (de um qualquer ano) e um outro com informação sobre a população dos vários países, pretendemos gerar uma série de gráficos de barras. A saber:

- Estudantes Erasmus por país;
- Estudantes Erasmus na proporção da população do país;
- Linguagem utilizada nos intercâmbios de estudo;
- Número de ECTS médio realizado pelos estudantes de cada país.

Todos os gráficos deverão aparecer ordenados por ordem decrescente. O aspeto geral dos gráficos deverá ser semelhante ao da figura abaixo.



A função principal deverá ter a seguinte assinatura:

```
def erasmus(erasmus_csv, paises_csv, max_paises_por_grafico)
```

onde `erasmus_csv` é o nome de um ficheiro em formato CSV com os dados sobre o programa Erasmus referentes a um dado ano, `paises_csv` é o nome de um ficheiro em formato CSV com os dados sobre a população mundial e `max_paises_por_grafico` é o número máximo de países a apresentar em cada gráfico. Assuma que ambos os ficheiros existem e podem ser lidos do disco.

O vosso programa deve ler ficheiros CSV descarregados dos sítios mencionados acima, e para uma lista de dicionários:

- O ficheiro com as populações não deve ser alterado;
- O ficheiro com os dados erasmus deve estar no formato de 2012–2013. Para testar a vossa solução com dados de outros anos poderão ter de alterar os nomes de algumas colunas, de modo a tornar o ficheiro CSV conforme o formato de 2012–2013.

Exeptuando o caso acima, não deve alterar o conteúdo dos ficheiros antes de os utilizar no seu programa. A primeira linha do ficheiro descarregado determina os nomes das chaves do dicionário. São estas as chaves que os vossos programas devem utilizar. Não devem em caso algum alterar ou converter estas chaves (para obter, por exemplo, nomes de chaves mais legíveis ou em português). Não devem também trocar as vírgulas por ponto e vírgulas antes de lerem o ficheiro CSV. Tal permite testar todas as soluções de um modo uniforme.

Além disso, *não* deverá assumir que os ficheiros contêm apenas dados válidos. O mais provável é conterem dados desconformes: no sítio onde um número é esperado podemos encontrar uma cadeia de carateres que não representa um número. A função de leitura dos dados deverá substituir dados omissos ou inválidos por dados válidos. Além disso todas as strings que ocorram nas células do ficheiro (e relevantes para a preparação dos gráficos) devem ser normalizas para letra maiúscula.

A data a figurar na figura a gerar (`2012–2013` no caso apresentado) deverá ser lida do ficheiro `erasmus_csv`, coluna `'STUDY_START_DATE'`. De um modo geral os anos letivos começam em setembro. Como haverá muitas linhas no ficheiro e algumas com datas inválidas, o ano inicial (`2012` no caso apresentado) é retirado da primeira data na coluna `'STUDY_START_DATE'` cujo mês seja setembro. Deste modo a função `erasmus` poderá ser utilizada com dados referentes a anos diferentes.

Para além da função `erasmus`, o seu trabalho deverá conter

obrigatoriamente duas funções que esperamos contribuam para a função `erasmus`. São elas:

```
def acumula (lista_dicionarios, chave_pesquisa):
    """
    Requires: A chave_pesquisa ocorre em todos os
   dicionários constantes na lista_dicionarios.
    Ensures: Devolve um dicionario onde a) as várias chaves
   são os valores associados à chave_pesquisa na
   lista_dicionarios e b) os valores do resultado são o
   número de vezes que as chaves do resultado ocorrem na
   lista_dicionarios.

    >>> l = [{'x': 'a', 'y': 23}, {'x': 'b', 'y': 78},
   {'x': 'a', 'y': 99}]
    >>> acumula (l, 'x') == {'a': 2, 'b': 1}
    True
    """

 def soma (lista_dicionarios, chave_pesquisa, chave_soma):
    """
    Requires: A chave_pesquisa e a chave_soma ocorrem em
   todos os dicionários constantes na lista_dicionarios. O
   valor associado à chave_soma é um número.
    Ensures: Devolve um dicionario onde a) as várias chaves
   são os valores associados à chave_pesquisa na
   lista_dicionarios e b) os valores do resultado são
   obtidos pela soma dos valores associados à
   chave_pesquisa na lista_dicionarios.

    >>> l = [{'x': 'a', 'y': 23}, {'x': 'b', 'y': 78},
   {'x': 'a', 'y': 99}]
    >>> soma (l, 'x', 'y') == {'a': 122, 'b': 78}
    True
    """
```

Neste projeto não pedimos que indiquem a complexidade das vossas funções, mas não queremos esperar muito quando chegar à altura de testar as vossas soluções. Assim, o tempo de executar a função `erasmus` deve ficar a abaixo de um certo valor, quando corrido numa máquina dos laboratórios do Departamento de Informática. O tempo máximo será indicado durante a semana de 15 de maio de 2017.

Para cada função que incluir no seu trabalho, junte dentro do `docstring`:

- Uma descrição da função incluindo o seu contrato (na forma de

`Requires:` e `Ensures:`), tal como sugerido nas aulas.

Para além disso, para cada uma das funções `acumula` e `soma`, apresente:

- Testes *baseados na partição do espaço de entrada*, no formato `doctest`. Indique no `docstring` as características e as regiões escolhidas; por exemplo:

  ```
  Características e regiões:
  - lista vazia?: True, False
  - número de ocorrências ...: ...
  ```

  e para cada teste a combinação das regiões que lhe deu origem:

  ```
  >>> funcao({'mae': 123})  # não vazia, 1, ...
  ```

Para além disso, o módulo em si deve estar equipado com uma descrição em formato `docstring`, tal como sugerido nas aulas. Não se esqueça de incluir o seu nome e número de estudante:
`__author__ = Maria Lopes, 45638.`

Tome em especial atenção os seguintes pontos.

- O nome do ficheiro que contém o vosso trabalho deve ser chamado `trabalho4_XXXXX.py` onde `XXXXX` é o vosso número de estudante.
- A primeira linha deste ficheiro deverá ser `# -*-coding:utf-8-*-`.
- O vosso código será testado por um processo automatizado. É indispensável que as três funções se chamem exatamente `erasmus`, `acumula` e `soma` e que esperem o número e tipo de parâmetros mencionados.
- Este é um trabalho de resolução em *grupos de dois alunos*. Os trabalhos devem ser entregues no Moodle até às 23:59 do dia 25 de maio de 2017.
- Os trabalhos de todos os alunos serão comparados por uma aplicação computacional. Releia com atenção a sinopse e lembre-se: "Alunos detetados em situação de fraude ou plágio, plagiadores e plagiados, ficam reprovados à disciplina (sem prejuízo de ser acionado processo disciplinar concomitante)".