# Programação II

# Projeto de programação

Crime numa grande cidade

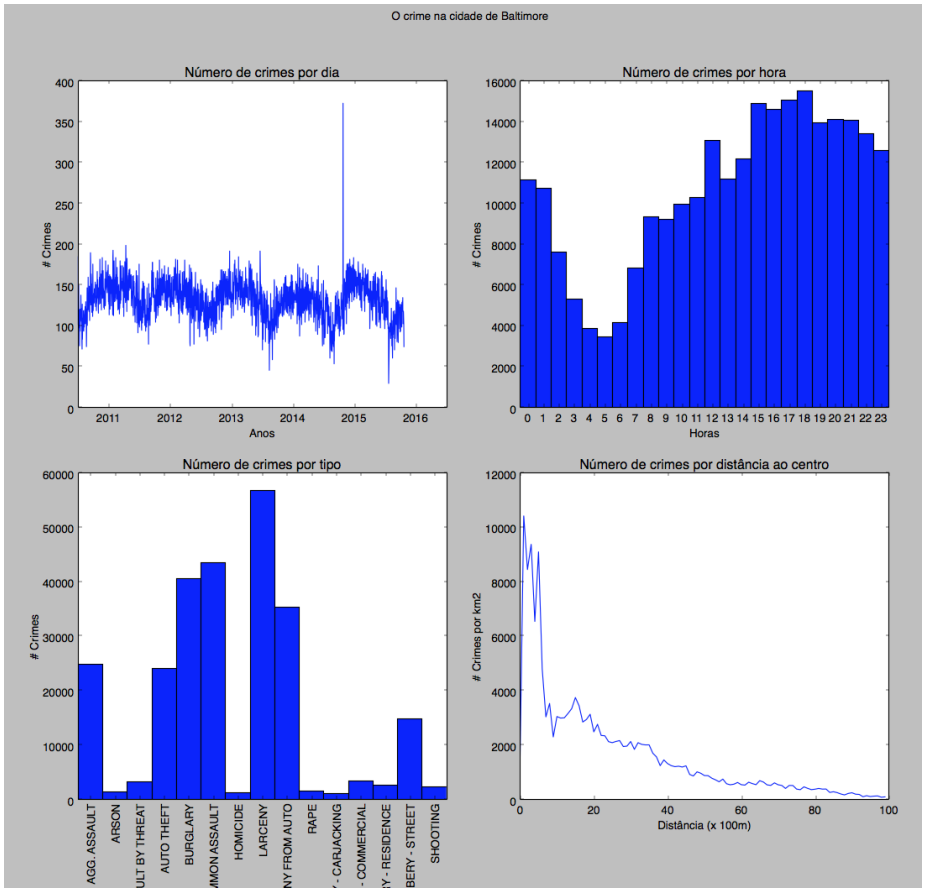
Entrega a 29 de maio de 2016

O crime nas cidades é tema que nos preocupa a todos. Perceber o padrão do crime numa larga escala temporal pode ajudar os decisores a tomar medidas para minorar o problema. E nada como gráficos para ajudar a visualizar a informação.

Este trabalho pretende analisar o fenómeno do crime numa grande cidade, utilizando para isso dados públicos da cidade de Baltimore nos Estados Unidos da América. O *Data Search Baltimore*🏹 publica todo o tipo de dados sobre Baltimore. Baseado nestes dados procuramos os seguintes indicadores:

1. O número de crimes participados em função do tempo. Este gráfico ajuda a perceber qual a tendência do crime ao longo do tempo. Está estável ou tem vindo a aumentar/diminuir? Haverá mais ocorrências de crimes no verão do que no inverno?
2. O número de crimes participados em função da hora do dia. Este gráfico ajuda a perceber as horas do dia em que há maior incidência de crimes. Será de noite?
3. O número de crimes participados em função do tipo de crime. Este gráfico permite perceber o tipo de crime prevalecente. Será furto de automóveis ou assaltos de rua?
4. O número de crimes por quilómetro quadrado em função da distância ao centro da cidade. Este gráfico permite perceber como se comporta o número de crimes à medida que nos afastamos do centro da cidade. Haverá mais crime na periferia?

O resultado deverá ser apresentado na forma de **uma** figura com quatro gráficos semelhante à apresentada a seguir. Tenha particular atenção às etiquetas das abcissas dos primeiro e último gráficos: no primeiro pretendemos que apareçam apenas os anos, no último queremos etiquetas de




2 em 2 quilómetros. Para além disso, o último gráfico deverá apresentar apenas dados para os primeiros 10km.

Os dados para as estatísticas devem ser obtidos em *Data Search Baltimore Victim-based Crime Data*. No canto superior direito da página encontra oito separadores coloridos. Seleccione o azul, com etiqueta "Export", e descarregue um ficheiro CSV com a informação relevante.

Escreva as seguintes funções.

1. Uma função `crimes(nome_ficheiro)` que recebe o nome de um ficheiro como parâmetro, e que traça uma figura com quatro gráficos, tal como explicado acima.

   **Nota muito importante**: A função deve ler um ficheiro CSV *tal como descarregado* do sítio *Data Search Baltimore Victim-based Crime Data* para uma lista de dicionários (a que chamamos uma tabela). Não deve em caso algum alterar o conteúdo deste ficheiro antes de o importar 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.

Para além desta função, prepare quatro funções todas com a mesma assinatura: recebem uma lista de dicionários com a informação sobre os crimes (lida diretamente de um ficheiro CSV descarregado do *Data Search Baltimore*, e inalterada, tal como descrito acima) e que devolvem pares `(abcissas, ordenadas)` para cada um dos gráficos. Estas funções **não** devem apresentar os gráficos, destinando-se apenas a preparar os dados para serem depois visualisados como gráficos.

2. A função `crimes_por_data(tabela_crimes)` recebe uma lista de dicionários e devolve um par de listas: abcissas e ordenadas. As abcissas contêm as datas dos crimes por ordem crescente; as ordenadas contêm o número de crimes que ocorreram em cada data. As datas tomam o formato `"AAAAMMDD"`, por exemplo, `"20160529"`. Tome atenção que na visualização do gráfico deverá apresentar a informação organizada pelos anos para os quais existem registos no CSV, sendo que a largura de cada ano no eixo dos *X* corresponderá ao total de dias deste (365 ou 366 dias), mesmo que para alguns dias não existam registos no ficheiro CSV. Sugestão: utilize o módulo `datetime` para trabalhar com datas, por exemplo, para converter datas em formato *string* em valores do tipo `date`, e calcular a diferença de dias entre valores deste último tipo.

3. A função `crimes_por_hora(tabela_crimes)` recebe uma lista de dicionários e devolve um par de listas: abcissas e ordenadas. As abcissas contêm uma lista ordenada de horas entre `0` e `23`; as ordenadas contêm o total de crimes, independentemente da data de ocorrência destes, nas horas correspondentes na lista das abcissas. Considere crimes registados às `24` horas como tendo ocorrido às `0` horas. Tome atenção que na visualização do gráfico deverá apresentar todas as horas, mesmo aquelas para as quais não existem registos de crime.

4. A função `crimes_por_tipo(tabela_crimes)` recebe uma lista de dicionários e devolve um par de listas: abcissas e ordenadas. As abcissas contêm o tipo do crime, dado por um valor do tipo *string* constante no campo *Description* do ficheiro CSV); as ordenadas contêm o número de crimes que ocorreram para o tipo correspondente.

5. A função `crimes_por_distancia(tabela_crimes)` recebe uma lista de dicionários e devolve um par de listas: abcissas e ordenadas. As abcissas contêm distâncias múltiplos de 100m ao centro da cidade, i.e. `[0,1,2,...]`, sendo que a primeira representa a coroa circular dos 0m

(inclusivé) aos 100m (exclusivé), a segunda representa a coroa circular dos 100m (inclusivé) aos 200m (exclusivé), e assim sucessivamente. As ordenadas contêm a densidade dos crimes em cada coroa circular, i.e., *o número de crimes por quilómetro quadrado*, pela ordem correspondente da lista das abcissas. Por exemplo, se $n$ for o número de crimes ocorridos dentro da terceira coroa, i.e., entre os 200m (inclusivé) e os 300m (exclusivé), a respetiva densidade de crimes será calculada dividindo $n$ pela área desta coroa em quilómetros quadrados, sendo esta última dada pela fórmula $\pi \times 0.3^2 - \pi \times 0.2^2$, e arredondando o resultado para o inteiro mais próximo. Importe o módulo `math` e use o valor de `math.pi` como o valor de $\pi$. Considere o centro da cidade situado nas coordenadas GPS $(39.289444, -76.616667)$, e ignore as entradas na tabela que não contêm coordenadas GPS. O código Python anexo a este enunciado contém uma função para calcular a distância entre duas localizações GPS.

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 carregar o ficheiro CSV somado ao de calcular as quatro funções `crimes_por_X` 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 9 de maio de 2016.

Na página da disciplina encontram um ficheiro Python com um esqueleto das funções principais, incluindo a função `haversine` para calcular a distância entre dois pontos dados por coordenadas GPS, e a função `go_time` que permite verificar o tempo de carregar o ficheiro CSV e calcular as quatro funções `crimes_por_X`.

Para cada função que considerar necessária (incluindo as cinco funções acima enumeradas), inclua os seguintes elementos.

- Uma descrição no formato *docstring*, tal como sugerido nas aulas.
- Um contrato usando cláusulas `Requires` e `Ensures`, também no formato *docstring*.

O módulo em si deve também estar equipado com uma descrição em formato *docstring*. Não se esqueçam de incluir os vossos nomes e números de estudante:

```
__author__ = "Ana Silva, 45638; Luís Pedro, 43231".
```

Durante o processo de desenvolvimento, preste atenção aos seguintes pontos.

- O código está bem estruturado, decomposto em várias funções?
- Cada função tem um objectivo simples e bem definido?
- As funções não são grandes demais? (isto é, fazem demasiadas coisas)

- O *docstring* e os contratos estão claramente enunciados, em português escorreito?
- Há código duplicado? (ou será que posso escrever uma função que evita a duplicação?)
- Será que a utilização duma lista em compreensão, duma função de ordem superior, ou duma outra função da biblioteca da linguagem Python, tornará o código mais legível ou mais compacto?

Este é um trabalho de resolução em *grupos de dois alunos*. Os trabalhos devem ser entregues no Moodle até às 23:55 do dia 29 de maio de 2016.

Os trabalhos de todos os alunos serão comparados por uma aplicação computacional. 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)".