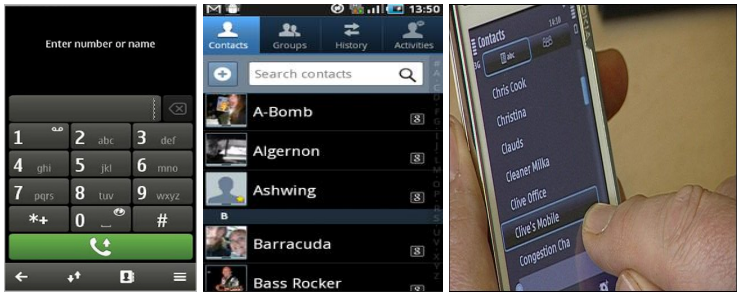
Programação II

Trabalho de casa 1

Esqueci o número de telefone

Entrega a 20 de março de 2017

Alguns de vós já ouviram falar em telefones móveis e a capacidade incrível que estes têm de associar nomes a números de telefone. Esqueceram o número de telefone da mãe? (ou será que nunca o souberam?). Não faz mal: os telemóveis mais recentes vêm equipados com a possibilidade de procurar números de telefone a partir de nomes.



Para este exercício vamos assumir que a informação referente aos números de telefone está guardada num dicionário cujas chaves são os nomes e os valores são os números de telefone. Imaginem o seguinte dicionário:

```
{'mae': 123, 'miguel adao': 987, 'maria lopes': 567,
 'pai': 468}
```

Falar para a mãe é fácil: basta escrever a palavra `mae` e automagicamente o número `123` aparece no visor do telefone. E se quiserem telefonar para a Maria Lopes? Terão de bater todas as letras do nome, incluindo o espaço? Felizmente que o nível tecnológico dos telefones (leia-se o software) permite uma solução alternativa.

Quando batemos `m` aparece uma lista *ordenada* de todas as entradas na lista de contactos cujo nome começa por `m`: neste caso uma lista com três entradas. Avançamos para o `a` e a lista fica restrita a duas entradas: `'mae'` e `'maria lopes'`. No nosso exemplo, à terceira letra ficamos com uma lista com `'mae'` ou com `'maria lopes'` ou então com uma lista vazia, dependendo da tecla que batermos.
O propósito do primeiro trabalho é implementar este tipo de funcionalidade. Tal será conseguido através da escrita de duas funções e dos respectivos testes, tal como explicado abaixo.
Para concretizar este cenário precisamos de uma função que, dado um dicionário e uma `string` obtém o subdicionário cujas chaves começam com a `string`. Infelizmente esta função não resolve completamente o problema: é que para nós a ordem é importante (queremos que `mae` apareça antes de `maria`), mas dicionários não podem ser ordenados. Por exemplo:

```
>>> {'mae': 123, 'maria lopes': 567}
{'maria lopes': 567, 'mae': 123}
```

Assim sendo, precisamos de uma segunda função que transforme o dicionário numa estrutura de dados que possa ser ordenada. Vamos escolher uma lista de pares da forma `(nome, numero)`. Por exemplo:

```
>>> pares_ordenados({'maria lopes': 567, 'mae': 123})
[('mae', 123), ('maria lopes', 567)]
```

Em resumo, precisamos de duas funções:

- `subdicionario(dicionario, palavra)`
- `pares_ordenados(dicionario)`

Nota. Dado que os dicionários não são impressos por nenhuma ordem em particular, não podemos escrever testes deste tipo:

```
>>> subdicionario(..., ...)
{'mae': 123, 'maria lopes': 567}
```

Em vez disso tomamos partido da capacidade do Python de comparar dicionários e escrevemos:

```
>>> subdicionario(..., ...) == {'mae': 123, 'maria lopes':
    567}
True
```

Para cada uma das duas funções, inclua dentro do `docstring`:

1. Uma descrição da função incluindo o seu contrato (na forma de `Requires:` e `Ensures:`), tal como sugerido nas aulas.

2. 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:

   ```
   >>> pares_ordenados({'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.

- Pode utilizar qualquer função da biblioteca Python, exceto as funções que obtêm as chaves, os valores e os itens de um dicionário.
- Para ordenar pode utilizar as funções `sort` ou **`sorted`**.
- O vosso código será testado por um processo automatizado. É indispensável que as vossas funções se chamem exatamente `pares_ordenados` e `subdicionario` e que esperem exatamente o número de parâmetros enunciados acima.
- Este é um trabalho de resolução individual. Os trabalhos devem ser entregues no Moodle até às 23:59 do dia 20 de março 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)".