Programação II

Trabalho 3

Testes de funções e programação funcional

Entrega a 5 de abril de 2021

Sistema de planeamento de itinerário Neste trabalho pretende-se implementar funções que permitam planear itinerários que passem através de vários pontos geográficos. Considere que dispõe de um dicionário com informação sobre a localização de várias cidades portuguesas em coordenadas GPS. Este dicionário está disponível num ficheiro `cidades.py` que pode encontrar na página Moodle da cadeira. Eis um excerto.

```
cidades = {'Lisboa': (38.7452, -9.1604),
           'Vila Nova de Gaia': (41.1333, -8.6167),
           'Porto': (41.1495, -8.6108),
           ...
          }
```

A partir das coordenadas GPS é possível calcular a distância entre duas cidades. Para simplificar, vamos assumir que, em Portugal continental, 1 grau em latitude (primeira coordenada) corresponde a 111,1949km, que 1 grau em longitude (segunda coordenada) corresponde a 85,1102km, e que a distância entre dois pontos é dada pela fórmula cartesiana, isto é, que a terra é plana em Portugal continental. Por exemplo, sabemos que Lisboa tem coordenadas (38,7452, -9,1604) e que o Porto tem coordenadas (41,1495, -8,6108), pelo que a distância entre as duas cidades deverá ser aproximadamente 271,407km. Lembre-se que a linguagem Python fornece a função `math.sqrt` no módulo `math` para calcular raízes quadradas.

Um itinerário corresponde a uma lista de cidades que indica a ordem pela qual estas deverão ser visitadas. Por exemplo:

```
itinerario = ['Lisboa', 'Setúbal', 'Coimbra', 'Aveiro'
    , 'Viseu', 'Porto']
```

Assuma que um itinerário tem sempre pelo menos duas cidades (não se esqueça de juntar os pressupostos de cada função no `docstring`).

Funções de ordem superior Para este trabalho deverá escrever as seguintes funções, recorrendo ao que aprendeu sobre programação funcional e funções de ordem superior.

1. Uma função `distancia_itinerario(itinerario)` que calcula a distância total de um itinerario (em km), ou seja, a soma das distâncias entre duas cidades consecutivas. Exemplo de execução:

   ```
   >>> round(distancia_itinerario(['Lisboa', 'Setúbal
       ', 'Coimbra', 'Aveiro', 'Viseu', 'Porto']), 3)
   419.256
   ```

2. Uma função `adicionar_cidade(itinerario, cidade)` que adiciona uma cidade a um itinerário já existente. Esta cidade deve ser colocada entre duas cidades do itinerário original, de modo a minimizar o desvio adicional. As cidades inicial e final do itinerário não deverão ser alteradas. No exemplo de execução que se segue, vemos que o melhor momento para visitar Aveiro é após termos passado por Viseu, mas antes de chegar ao Porto.

   ```
   >>> adicionar_cidade(['Lisboa', 'Setúbal', '
       Coimbra', 'Viseu', 'Porto'],  'Aveiro')
   ['Lisboa', 'Setúbal', 'Coimbra', 'Viseu', 'Aveiro'
       , 'Porto']
   ```

3. Uma função `construir_itinerario(origem, destino, lista_cidades)` que constrói um itinerário a partir de uma lista de cidades, usando a função do passo anterior. A sua implementação deverá adicionar sequencialmente cada cidade da lista dada ao itinerário, de forma a minimizar a distância percorrida. As cidades inicial (`origem`) e final (`destino`) do itinerário não deverão ser alteradas. No exemplo de execução que se segue, começamos por adicionar Viseu entre Lisboa e o Porto; depois, colocamos Coimbra entre Lisboa e Viseu; de seguida, Aveiro é inserida entre Viseu e o Porto; finalmente, colocamos Setúbal entre Lisboa e Coimbra:

```
>>> construir_itinerario('Lisboa', 'Porto', ['
    Viseu', 'Coimbra', 'Aveiro', 'Setúbal'])
['Lisboa', 'Setúbal', 'Coimbra', 'Viseu', 'Aveiro'
    , 'Porto']
```

Testes Para cada uma das três funções assinaladas deverá incluir, para além do `docstring`, uma bateria de testes, utilizando a técnica da partição do espaço de entrada. Deverá seguir os seguintes passos.

1. Modelar o domínio da função, identificando **entre duas e quatro** características.
2. Para cada característica, identificar blocos adequados.
3. Combinar todos os blocos, eliminando as combinações inviáveis. Apresente, num comentário Python, os seus resultados numa tabela da seguinte forma:

Características e blocos			Testes	
Característica 1	…	Característica n	Argumentos	Resultado
…	…	…	…	…

   Deverá obter **entre cinco e quinze** combinações viáveis.
4. Apresentar os testes em formato `doctest`.

Especial atenção Tome em especial atenção os seguintes pontos.

- Para obter nota máxima, deverá adoptar uma implementação exclusivamente usando programação funcional e funções de ordem superior (**`map`**, **`filter`**, **`reduce`** e afins). **Por cada utilização de listas por compreensão, ciclos `for` / `while` ou recursão será descontado `10%` da nota final**!
- As regras de boas práticas de desenvolvimento de software apontam para um número máximo de cerca de 10 linhas por função. Identifique as abstrações relevantes e implemente cada uma numa função separada.

- Pode utilizar e adaptar funções que tenham sido apresentadas nas aulas.
- Cada função que escrever (as obrigatórias `distancia_itinerario`, `adicionar_cidade`, `construir_itinerario`, bem como aquelas que porventura achar relevantes) deve estar equipada com uma descrição em formato `docstring`, tal como sugerido nas aulas.
- Pode e deve incluir testes para as funções adicionais que implementar; no entanto, estes não são obrigatórios.
- Não se esqueça de incluir o seu nome e número de estudante no início do ficheiro: `__author__ = Maria Lopes, 45638`.
- O vosso código será testado por um processo automatizado. É indispensável que este contenha as funções com os nomes `distancia_itinerario`, `adicionar_cidade`, `construir_itinerario`, que a função espere exatamente o número e o tipo dos parâmetros e que devolva exatamente um objeto do tipo de dados pedido.
- Este é um trabalho de resolução individual. Os trabalhos devem ser entregues no Moodle até às 23:59 do dia 5 de abril de 2021.
- Os trabalhos de todos os alunos serão comparados por uma aplicação de deteção de plágio em programas. Recorde o seguinte texto na secção Integridade Académica da Sinopse:

  "Alunos detetados em situação de fraude ou plágio (plagiadores e plagiados) em alguma prova ficam reprovados à disciplina e serão alvo de processo disciplinar, o que levará a um registo dessa incidência no processo de aluno, podendo conduzir à suspensão letiva."