Sequências

1. **Antes da aula**: Tente prever o que acontece nestes excertos no PythonTutor:

   a) Revisão de listas.
   b) Revisão de tuplos.
   c) Revisão de strings.

2. O programa `elevador.py` tem uma lista com o número de moradores em cada piso de um prédio. Queremos calcular a distância que o elevador percorre todos os dias.

   a) Crie uma função `distanciaElevador` que calcule e devolva esse valor. Note que cada andar tem uma altura de 3 m e cada morador faz 4 viagens por dia entre o R/C (piso 0) e o seu andar. Experimente o programa alterando os números de moradores.

   b) Complete a função main para calcular a distância (km) percorridos num ano.

   c) Crie uma função para pedir ao utilizador o número de pisos de outro prédio e o número de moradores em cada piso e use-a na função `main` para pedir esses dados e calcular a distância nesse prédio.

```
Quantos pisos tem o prédio? 3
Quantos moradores no piso 0? 4
Quantos moradores no piso 1? 5
Quantos moradores no piso 2? 5
No predio [4, 5, 5]
o elevador percorre 180.0 m por dia
```

3. Siga os seguintes passos, testando cada um:

   a) Crie uma função `inputFloatList()` que leia uma sequência de números introduzidos pelo utilizador e os devolva numa nova lista. O utilizador deve introduzir um número por linha e indicar o fim da lista com uma linha vazia.

   b) Crie uma função `countLower(lst, v)` que conte (e devolva) quantos elementos da lista `lst` são inferiores ao valor `v`.

   c) Crie uma função `minmax(lst)` que devolva o mínimo e o máximo de uma lista de valores. Consegue fazê-la sem usar as funções `min`, `max`, `sort`, nem `sorted`?

   d) Recorra às funções anteriores para fazer um programa que leia uma lista de números, determine o valor médio entre o mínimo e o máximo e conte quantos números são inferiores a esse valor.

4. Escreva uma função que, dada uma lista de equipas de futebol, crie e devolva uma lista de todos os jogos que se podem fazer entre elas. Por exemplo:

```
allMatches(["FCP", "SCP", "SLB"]) ->
  [('FCP','SCP'), ('FCP','SLB'), ('SCP','FCP'), ...]
```

   Com 3 equipas deve obter 6 jogos, com 4 equipas deve obter 12 jogos. Confirme e teste com ainda mais equipas. (E teste no CodeCheck.)

5. Crie uma função que conte quantos dígitos há numa dada string. Por exemplo: `countDigits("23 mil 456")` deve devolver `5`. *Sugestão: o método `isdigit` verifica se uma string só tem dígitos, e.g., `"2".isdigit() -> True`.* (Teste no CodeCheck.)

6. Crie uma função que, dado um nome, crie uma versão abreviada, formada apenas pelas letras maiúsculas desse nome. Por exemplo:

   ```
   shorten("Universidade de Aveiro") -> "UA",
   shorten("United Nations Organization") -> "UNO".
   ```

   Sugestão: o método `str.isupper` verifica se uma string só tem maiúsculas, e.g., `"A".isupper() -> True`.

7. Crie uma função `ispalindrome(s)` que devolva um valor booleano indicando se a string `s` é um palíndromo ou não.

8. Resolva os exercícios abaixo usando o sistema CodeCheck para as testar.

   a) Crie uma função que, dada uma string, devolve outra composta pelos caracteres das posições pares seguidos pelos caracteres das posições ímpares da primeira. Por exemplo, `evenThenOdd("abcd")` deve devolver `"acbd"`. Pode fazê-lo usando *slicing* e concatenação. (CodeCheck).

   b) Crie uma função que, dada uma string `s`, devolve uma string semelhante mas sem caracteres adjacentes duplicados. Por exemplo, para o argumento `"Mississippi"` deve devolver `"Misisipi"`. (CodeCheck).

   c) Crie uma função que, dado um inteiro não negativo `n`, devolve uma lista contendo 1, 2, 2, 3, 3, 3, 4, 4, 4, 4, ... e finalmente `n` repetido `n` vezes. (CodeCheck).

   d) Crie uma função que, dada uma lista de valores, devolve o índice da primeira ocorrência do maior valor. Pode admitir que a lista não está vazia. Não pode usar as funções `max`, `find` nem `index`. (CodeCheck).

   (Estes exercícios foram criados por Cay Horstmann, professor na San Jose State University e autor de diversos livros sobre programação.)

9. No programa `dates5.py`:

   a) Complete a função `parseMDY(dateStr)` para devolver um terno de inteiros com o ano, mês e dia de uma data escrita no formato americano "Mes/Dia/Ano". Por exemplo, `parseMDY("12/25/2024")` deve devolver `(2024, 12, 25)`. Se a data contiver apenas o ano, deve devolver (ano, 0, 0).

   b) Complete a função yearsBetween(d1, d2) para devolver o número inteiro de anos entre duas datas expressas como ternos (ano, mês, dia).

   Confira os resultados dos testes feitos na função principal.

10. O ficheiro `composers.txt` tem datas de nascimento e da morte de compositores famosos. O programa `composers.py` lê os dados desse ficheiro[1] para uma lista e depois mostra-os. Analise e corra o programa para ver como funciona.

    a) Modifique a função load_lifetimes_file para guardar os dados na forma de ternos (data1, data2, nome), onde as datas são ternos na forma (ano, mês, dia). No ficheiro, as datas e o nome estão separados por Tabs (`"\t"`).

    b) Altere o programa para mostrar o nome, idade e data da morte de cada compositor em colunas devidamente alinhadas.

    c) Altere o programa para mostrar apenas compositores nascidos no século XIX.

    Reutilize as funções criadas no exercício anterior. Até pode importá-las desse módulo com uma instrução `from dates5 import parseMDY, yearsBetween`.

11. O programa `telephones.py` define duas listas, uma com números de telefone e outra com os nomes correspondentes.

```
telList = ['975318642', '234000111', '777888333', ...]
nameList = ['Angelina', 'Brad',      'Claudia',   ...]
```

    a) Complete a função `telToName` que, dado um número de telefone (e as duas listas), devolve o nome respetivo (ou o próprio número, se não estiver na lista). Isto é o que os telemóveis fazem quando recebem uma chamada.

    b) Complete a função `nameToTels` que, dada parte de um nome, devolve a lista dos números correspondentes a nomes que incluem essa parte. (Como quando pesquisa na lista de contactos do telemóvel.)

    c) Corra o programa para testar essas funções.

[1] Os detalhes da leitura de ficheiros serão tratados numa aula futura.