

Sequências

Exercícios

1. Tente adivinhar o resultado de cada uma das instruções abaixo. Algumas não têm resultado e outras dão erros. Use o Python em modo interativo para confirmar. (Veja se acerta mais que o/a colega do lado!)

| | |
|------------------------------------|---|
| <code>lst = [5, 3, 8, 7]</code> | <code>t = ("ana", (1974, 4, 25))</code> |
| <code>len(lst)</code> | <code>len(t)</code> |
| <code>lst[2]</code> | <code>t[1]</code> |
| <code>lst[4]</code> | <code>t[1][0]</code> |
| <code>lst[-4]</code> | <code>(1974, 3, 30) < (1974, 4, 1)</code> |
| <code>lst[1:3]</code> | <code>(1974, 3) < (1974, 3, -1, -2)</code> |
| <code>lst[:-1]</code> | <code>s = "abcde"</code> |
| <code>lst[2:2]</code> | <code>s[2:] + s[:2]</code> |
| <code>lst[2:2] = [99] ; lst</code> | <code>s[1::2]</code> |
| <code>lst.append(33) ; lst</code> | <code>s > "abel"</code> |

2. Siga os seguintes passos, testando cada um:

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

3. 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', ...]
```

- 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.
 - 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.)
 - Corra o programa para testar essas funções.
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. (Ou faça 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`.* ([CodeCheck](#).)
6. Crie uma função que, dado um nome, crie uma versão abreviada, formada apenas pelas letras maiúsculas. Por exemplo:

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

Sugestão: o método `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.)