8.1 Modifique a função `procurar` da aula téorica 13 que procura um texto num ficheiro para *contar o número de linhas* em que o texto ocorre. Exemplo (da aula teórica): `contar("armas", "luisadas_CantoI.txt")` dá 8

8.2 Escreva um programa que lê o ficheiro do Canto I dos *Lusíadas* (na página da UC) e procura o comprimento da palavra mais longas.
Sugestão: utilize o método `split()` das cadeias de carateres para partir uma linha numa lista de palavras.

8.3 Escreva um programa que lê o ficheiro do Canto I dos *Lusíadas* (na página da UC) e determina o *comprimento médio* das palavras do texto: a soma dos comprimentos de cada palavra a dividir pelo número de palavras. 1 *Sugestão*: utilize o método `split()` das cadeias de carateres para partir uma linha numa lista de palavras.

▷ **8.4** Considere a *sequência de Collatz* [1]: começamos com um inteiro positivo n dado; cada novo valor é obtido a partir do anterior:

- se n é par: divimos n a metade e continuamos;
- se n é ímpar: multiplicamos n por 3, somamos 1 e continuamos;
- terminamos quando $n = 1$.

Exemplo: para valor inicial $n = 6$ os valores gerados são $6 \to 3 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$.

Escreva uma função `collatz(n)` que determina esta sequência para um inteiro dado; o resultado deve ser a lista dos valores gerados. Exemplo:

```
>>> collatz(6)
[6, 3, 10, 5, 16, 8, 4, 2, 1]
```

8.5 Usando a função `collatz(n)` do exercício anterior, defina um programa que escreve um ficheiro de texto `"tabela.txt"` com a tabela dos valores de n e do *comprimento* de `collatz(n)` de 1 até 1000; as primeiras linhas são:

```
n       len(collatz(n))
1       1
2       2
3       8
4       3
5       6
6       9
```

Use um programa externo (como o *Gnuplot*, *Excel* ou *Libre Office*) para traçar o gráfico destes pontos.

[1] Provar que este processo termina ou não para *todo* $n > 0$ é ainda um problema matemático em aberto. Mais informação em `http://en.wikipedia.org/wiki/Collatz_conjecture`

8.6 Escreva uma função `factorial(n)` para calcular o factorial de n e que lance uma exceção `TypeError` se n não for inteiro e `ValueError` se n for inteiro mas negativo. Sugestão: pode obter o tipo de uma variável x usando `type(`x`)`.

▷ **8.7** Escreva uma função `media(valores)` cujo resultado é a média aritmética de uma lista de valores. Se o argumento não for uma lista deve lançar uma exceção `TypeError`; se o argumento for a lista vazia deve lançar uma exceção `ValueError`.

▷ **8.8** Duas palavras ou frases são *anagramas* se se escrevem com as mesmas letras usadas o mesmo número de vezes mas eventualmente em posições diferentes. Por exemplo, a frase em Latim "Quid est veritas?" (*O que é a verdade?*) é um anagrama de "Est vir qui adest" (*É o homem que está diante de si*).

Escreva uma função `anagramas(txt1,txt2)` que verifique se duas cadeias são anagramas; o resultado deve ser `True` ou `False`. Deve considerar equivalentes as letras maiúsculas e minúsculas e ignorar todos os caracteres que não são letras (espaços, sinais de pontuação, etc.); pode ainda assumir que as cadeias não têm letras com acentos.

8.9 O código Morse associa cada letra do alfabeto a uma sequência de "pontos" e "traços", conforme a tabela seguinte:

A	`.-`	B	`-...`	C	`-.-.`	D	`-..`	E	`.`	F	`..-.`
G	`--.`	H	`....`	I	`..`	J	`.---`	K	`-.-`	L	`.-..`
M	`--`	N	`-.`	O	`---`	P	`.--.`	Q	`--.-`	R	`.-.`
S	`...`	T	`-`	U	`..-`	V	`...-`	W	`.--`	X	`-..-`
Y	`-.--`	Z	`--..`								

Escreva uma função `morse(txt)` que converte as letras numa sequência de carateres para Morse; o resultado deve ser uma cadeia com pontos e traços; use um espaço para separar sequências correspondentes às letras. Os carateres do texto original que não forem letras maiúsculas devem ser ignorados. Exemplos:

```
>>> morse('ABC')
'.- -... -.-.'
>>> morse('ATTACK AT DAWN')
'.- - - .- -.-. -.- .- - -.. .- .-- -.'
```

Sugestão: começe por definir a tabela de código Morse como um dicionário.

8.10 (T) O *Mastermind* é um jogo para duas pessoas que fazem de *codemaker* e *codebreaker*; o primeiro escolhe uma chave secreta (uma sequência de letras) e o segundo tenta deduzir a chave no menor número de tentativas. Por cada tentativa o *codemaker* diz duas pontuações: o *número de letras corretas na posição correta* e o *número de letras corretas mas na posição errada*.

Por exemplo, se a chave for `'acfb'` e a tentativa `'abcd'` as pontuações são (1, 2), ou seja, uma letra correta na posição correcta (`'a'`) e duas corretas mas em posições erradas (`'c'` e `'b'`).

Escreva uma função `pontua(chave,tentativa)` que, dado uma chave e tentativa, contabilize as duas pontuações acima; o resultado deverá ser um par com as pontuações. Tenha em atenção que as letras podem ocorrer repetidas. Por exemplo, `pontua('acfa','aacc')` deve dar `(1,2)`: um `'a'` na posição correcta e um `'a'` e `'c'` em posições erradas.