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

▷ **8.2** Escreva uma função `media(xs)` 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`.

8.3 Escreva uma função `longa(ficheiro)` que lê um ficheiro e procura a palavra de maior comprimento. Por exemplo:

```
>>> longa("lusiadas_CantoI.txt")
'Cristianíssima'
```

Sugestão: utilize o método `split()` das cadeias de carateres para partir uma linha numa lista de palavras.

8.4 Considere o exemplo da *sequência de Collatz* apresentada na aula teórica 11. Modifique a função `collatz(n)` para não imprimir os valores sucessivos de n e em vez disso calcular e retornar o *número de iterações executadas*.

Exemplos:

```
>>> collatz(1)
0
>>> collatz(4)
2
>>> collatz(7)
16
```

8.5 Considere a função `collatz(n)` do exercício anterior. Escreva um programa que escreve um ficheiro de texto `collatz.txt` com a tabela desta função para n de 1 até 1000. Use o programa `gnuplot` para desenhar um gráfico destes valores usando o seguinte comando:

```
gnuplot> plot "collatz.txt"
```

▷ **8.6** 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.

8.7 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.

8.8 (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.