Folha de apoio à aula prática 8 (exercícios para praticar antes da aula).

8.1 Escreva uma função para encontrar as raízes reais da equação quadrática $ax^2 + bx + c = 0$. O resultado deve ser uma lista vazia quando não há raízes reais, uma lista com um valor em vírgula flutuante quando há uma solução, e uma lista com dois valores quando há duas soluções.

8.2 Escreva uma função que, dada uma *string* com o nome completo de uma pessoa, devolva uma *string* com primeiro, segundo, e último nomes. Adicione um teste por forma a não contarem como nome as preposições `da das de do dos e`. Considere a possibilidade de o nome ter apenas uma ou duas palavras. (Sugestão: utilize `split` e `join`).

8.3 Um *quadrado mágico* é uma matriz de $n \times n$ que verifica as seguintes condições:

(a) é preenchida com números inteiros distintos de 1 até n^2;

(b) todas as linhas, colunas e diagonais somam o mesmo valor.

O exemplo seguinte representa um quadrado mágico em que cada linha, coluna e diagonal soma 15:

2	7	6
9	5	1
4	3	8

Num *quadrado pseudo-mágico* apenas a segunda condição é necessária.

Escreva uma função `pseudo_magico(A)` que testa se uma matriz (representada como uma lista de listas) é um *quadrado pseudo-mágico*; o resultado deve ser um valor lógico.

8.4 Crie uma lista contendo 1000 inteiros, aleatórios, com valores entre 0 e 100. Escreva a função `count_ints` que toma uma lista de valores numéricos como argumento e retorna um dicionário com o número de ocorrências de cada inteiro (se positivo). Teste-o com uma lista para as quais conheça o resultado e com a lista que criou nesta alínea.

8.5 Implemente a função `split(a)` que, dada uma lista de inteiros `a`, devolve um tuplo com duas listas: uma com os números pares existentes em `a`, e outra com os seus números ímpares. A ordem da lista `a` deve ser mantida.

8.6 A *sequência de Collatz* para um determinado valor $n > 0$ é obtida da seguinte forma: enquanto n não for igual a 1, gerar o próximo elemento da sequência usando a função:

$$f(n) = \begin{cases} n/2 & \text{se } n \text{ é par} \\ 3n+1 & \text{se } n \text{ é ímpar} \end{cases}$$

Escreva uma função `it_collatz(n)` que retorna o número de iterações executadas. Exemplos:

```
>>> it_collatz(1)
0
>>> it_collatz(7)
16
```

Construa um dicionário que associa a cada inteiro $i : 1 \leq i \leq 1000$ o valor de `it_collatz(i)`

8.7 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 a função `anagramas(txt1,txt2)` que verifica se as cadeias de carateres `txt1` e `txt2` são anagramas; o resultado deve ser `True` ou `False`. Deve considerar as letras maiúsculas e minúsculas equivalentes e ignorar todos os caracteres que não são letras (espaços, sinais de pontuação, etc.); pode ainda assumir não há carateres acentuados.

8.8 Um *triângulo de Sierpinski* de ordem 0 é um triângulo equilátero. Um triângulo de ordem 1 é constituido por 3 triângulos mais pequenos, ligeiramente separados na figura abaixo para facilitar a compreensão. A figura ilustra também triângulos de Sierpinski de ordem 2 e 3.

Defina uma função recursiva `sierpinski(n,lado)` para desenhar um triângulo de Sierpinski de ordem `n` e tamanho `lado`.

8.9 Defina a função `wordfreq(txt)` que, dada a *string* `txt` retorna uma lista com a(s) palavra(s) que aparece(m) nessa *string* mais vezes. Pode assumir que `txt` só contém palavras com maiúsculas ASCII separadas por espaços. Por exemplo, `wordfreq('NO NEWS IS NO GOOD NEWS')` deverá retornar `['NO', 'NEWS']`.

Sugestão: use dicionários associando a cada palavra o número de vezes que apareceu.

8.10 No campeonato nacional de futebol uma vitória conta 3 pontos, um empate 1 ponto e uma derrota 0 pontos. Em cada jogo, ganha a equipa que marcar mais golos, havendo um empate se o número de golos for o mesmo. Implemente a função `futebol(scores)` que retorna um dicionário com a pontuação de cada equipa. O parâmetro `scores` é uma lista de dicionários com os resultados da jornada. Cada dicionário tem nome de um clube na chave e o número de golos no valor. Por exemplo,

```
futebol([{"Vitória SC":2, "Boavista":1}, {"Gil Vicente":1, "Rio Ave":1},
         {"Famalicão":3, "Sporting":2}, {"FC Porto":0, "Benfica":0},
         {"Tondela":2, "Santa Clara":3}])
```

deverá retornar o dicionário

```
{'Vitória SC': 3, 'Boavista': 0, 'Gil Vicente': 1, 'Rio Ave': 1, 'Famalicão': 3,
 'Sporting': 0, 'FC Porto': 1, 'Benfica': 1, 'Santa Clara': 3, 'Tondela': 0}
```

8.11 Nos anos 50 do século passado, no campeonato de Formula 1 a pontuação de cada prova era atribuída com base na posição de cada corredor no final, de acordo com a seguinte tabela:

Posição	Pontos
1	8
2	6
3	4
4	3
5	2

Implemente a função `formula1(scores)` que, dada uma lista com a lista ordenada dos cinco melhor classificados, para cada prova de uma época, retorna um dicionário com a pontuação de cada corredor. Por exemplo,

```
formula1([
   ['Sainz', 'Verstappen', 'Hamilton', 'Ricciardo', 'Massa'],
   ['Bottas', 'Verstappen', 'Raikkonen', 'Stroll', 'Vettel'],
   ['Perez', 'Raikkonen', 'Verstappen', 'Hamilton', 'Vettel']
   ])
```

deverá retornar

```
{'Sainz': 8, 'Verstappen': 16, 'Hamilton': 7, 'Ricciardo': 3, 'Massa': 2,
 'Bottas': 8, 'Raikkonen': 10, 'Stroll': 3, 'Vettel': 4, 'Perez': 8}
```