4.1 A área A de um triângulo cujos lados medem a, b e c pode ser calculada usando a fórmula (atribuida ao matemático grego Heron)

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

onde $s = (a + b + c)/2$ é o semi-perímetro do triângulo. Implemente uma função `area_triangulo(a,b,c)` que calcule a área de um triângulo usando esta fórmula.

▷ **4.2** Escreva uma função `triangulo(a,b,c)` que classifica um triângulo como *equilátero*, *isósceles* ou *escaleno* dados os comprimentos a, b, c dos três lados; o resultado deve ser uma cadeia de carateres. Alguns exemplos:

```
>>> triangulo(3,3,3)
'equilátero'
>>> triangulo(3,2,3)
'isósceles'
>>> triangulo(3,4,5)
'escaleno'
```

▷ **4.3** Escreva uma função `classifica(p)` que, dada a pontuação p obtida num exame (de 0 a 100), retorna uma cadeia de carateres com a classificação; veja os exemplos e a tabela seguintes.

```
>>> classifica(55)
'suficiente'
>>> classifica(80)
'muito bom'
>>> classifica(110)
'inválido'
```

< 0 ou > 100	inválido
$\geq 0, < 50$	insuficiente
$\geq 50, < 70$	suficiente
$\geq 70, < 80$	bom
$\geq 80, < 90$	muito bom
$\geq 90, \leq 100$	excelente

4.4 Um ano é *bissexto* se for divisível por 4, exceto se for múltiplo de 100 e não for divisível por 400. Escreva a função `bissexto(n)` que retorna `True` se `n` for um ano bissexto e `False` no caso contrário.

4.5 Teste a função do exercício anterior (4.4) fazendo um programa que escreve uma tabela dos anos bissextos entre 2000 e 2020. Verifique os resultados com o calendário do computador.

▷ **4.6** Podemos contar algarismos decimais na representação de um número fazendo divisões inteiras por dez. Por exemplo: 9733 tem 4 algarismos porque podemos fazer quatro divisões inteiras sucessivas por 10 até obter quociente zero.
Escreva uma função `algarismos(n)` que retorna o número de algarismos decimais de n. Sugestão: utilize um ciclo `while`.

4.7 (T) O *menor divisor próprio* de um inteiro n é o menor inteiro d tal que $d > 1$ e d divide n (ou seja: o resto da divisão de n por d é zero).

(a) Escreva a definição duma função `mindiv(n)` que calcula o menor divisor próprio.

(b) Um inteiro n é *primo* se $n > 1$ e o menor divisor próprio de n é igual a n. Escreva uma definição da função `primo(n)` que testa se n é primo usando o critério anterior; o resultado deve ser `True` ou `False`.

(c) Note que se d é o menor divisor próprio de n e $d > \sqrt{n}$ então $d = n$ (porquê?). Modifique a definição da função da alínea (a) para usar esta propriedade para tornar mais rápida a pesquisa do menor divisor próprio.

4.8 A fórmula de Leibniz para aproximar π é:

$$\pi = 4 \times \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} + \cdots\right) = 4 \times \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1}$$

Implemente a função `leibniz(k)` que calcula o somatório dos primeiros `k` termos desta série. Documente a sua função com uma *docstring*.

4.9 Considere o programa para simular o lançamento de dois dados apresentado na aula teórica.

(a) Generalize este programa para uma função `soma2dados(k)` que estima a frequância da soma de dois dados dar um valor k.

(b) Use a função de alínea anterior para imprimir uma tabela da distribuição de frequência da soma de dois dados (entre 2 e 12).

4.10

As espirais da figura ao lado foram desenhadas usando o módulo *turtle* apenas mudando o ângulo de rotação entre cada segmento. Escreva um procedimento `espiral(...)` para desenhar espirais deste tipo.
Sugestão: pense quais os parâmetros que necessita para fazer o 1º desenho (ângulos retos) primeiro e depois generalize para caso geral.