3.1 Considere um programa que começa com a atribuição de uma lista de valores de temperatura em graus *Celsius* à variável `tempC`:

```
tempC = [-5,0,5,10,15,20,25]
```

(a) Escreva um ciclo `for` que imprime cada um dos valores de temperatura da lista `tempC` numa linha separada.

(b) Escreva um outro ciclo `for` que imprima os mesmos valores da lista `tempC`, mas gerados à custa da função `range`.

(c) Escreva um ciclo `while` que produza o mesmo resultado da alínea anterior.

(d) Escreva um ciclo em que para os valores de temperatura acima, em cada linha, imprime o valor em graus *Celsius* e o valor correspondente em graus *Fahrenheit*. Nota: Pode (e deve) utilizar a função `celsius(F)` implementada no exercício 1.14.

3.2 Utilizando a função `range`, escreva um programa que imprime os valores `10, 13, 16,..., 55`.

(a) Usando a instrução `break`, modifique-o para terminar o ciclo quando encontrar um múltiplo de 7.

(b) Usando a instrução `continue`, modifique-o para *não imprimir* valores múltiplos de 7.

3.3 Escreva uma função `media_arit(xs)` que, dada uma lista de n valores numéricos `xs`, retorna a média aritmética dos seus valores, isto é,

$$\frac{1}{n}\sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

3.4 Escreva uma função `media_geom(xs)` que, dada uma lista de n valores numéricos `xs`, retorna a média geométrica dos seus valores, isto é,

$$\left(\prod_{i=1}^{n} x_i\right)^{\frac{1}{n}} = \sqrt[n]{x_1 x_2 \cdots x_n}$$

3.5 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 resulta em `True` se `n` for um ano bissexto e `False` no caso contrário.

3.6 Teste a função do exercício anterior (3.5) fazendo um programa que escreve uma tabela dos anos bissextos entre 2000 e 2020. Verifique os resultados usando o calendário do computador.

3.7 Escreva uma definição da função `fatorial(n)` que, dado um inteiro positivo `n`, calcula e retorna o seu fatorial. Relembre que $n! = n \times (n-1) \times \cdots \times 1$.

3.8 Um número n diz-se um quadrado perfeito se, para algum natural k, pode ser escrito como a soma dos k primeiros números ímpares, isto é, $1+3+\cdots+k$. Os primeiros cinco quadrados perfeitos são $1, 4, 9, 16$ e 25. Escreva uma função `quadrado_perfeito(n)` cujo resultado é `True`, se n é um quadrado perfeito. Caso contrário, o resultado deverá ser `False`.

3.9 Um número n diz-se triangular se pode ser escrito como uma soma $1 + 2 + \cdots + k$ para algum natural k. Os primeiros cinco números triangulares são $1, 3, 6, 10$ e 15. Escreva uma função `triangular(n)` cujo resultado é `True` ou `False` conforme n é triangular ou não.

3.10 Podemos contar algarismos decimais na respresentação de um número fazendo divisões inteiras por dez. Por exemplo: 9733 tem 4 algarismos porque podemos fazer quatro divisões sucessivas por 10 obtendo os quocientes 973, 97, 9 e 0 (paramos quando chegamos a zero). Escreva uma função `algarismos(n)` cujo resultado é o número de algarismos decimais de n. Sugestão: use um ciclo `while` para repetir as divisões sucessivas e contar o número de iterações.

3.11 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 o seu menor divisor próprio for n. Escreva a definição duma função `primo(n)` cujo resultado é `True` ou `False` conforme n é ou não primo.

(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.

3.12 A fórmula de Leibniz para aproximar π é:

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

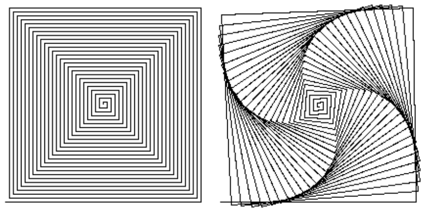
Implemente a função `leibniz(k)` que resulta no somatório dos primeiros `k` termos desta série. Documente a sua função com uma *docstring.*

3.13 O *coeficiente binomial* $\binom{n}{k}$, também designado por *combinações de n em k* pode ser calculado usado a seguinte fórmula (com $0 \leq k \leq n$):

$$\binom{n}{k} = \frac{n \times (n-1) \times \cdots \times (n-k+1)}{k \times (k-1) \times \cdots \times 1} = \frac{n!}{k!(n-k)!}$$

Escreva uma função `binom(n,k)` que calcule coeficientes binomiais.

3.14



As espirais da figura ao lado foram desenhadas usando o módulo *turtle* apenas mudando o ângulo de rotação entre cada lado. Escreva uma função `espiral(...)` para desenhar espirais deste tipo.