Bibliotecas NumPy e MatPlotLib

Tópicos

- Introdução da biblioteca **NumPy** que permite fazer operações com vetores
- Introdução da biblioteca **MatPlotLib** que permite traçar gráficos

Exercícios

1. O programa `plot.py` traça os gráficos de duas funções. Experimente executá-lo. Terá de fechar a janela para o terminar. Edite o programa para tentar percebê-lo. Pode imprimir valores das variáveis ou modificar alguns parâmetros para ver o que acontece.

    a) Altere o programa para adicionar um segundo gráfico ao *subplot(2, 1, 2)*, com o produto das funções y1 e y2. Trace o gráfico a verde.

    b) Adicione uma segunda figura com o gráfico do polinómio do 2.º grau, $x^2 + 3x + 1$, no intervalo [-3, 2] e com incrementos de x de 0,1. Desenhe a grelha *plt.grid()*..

2. O programa seno.py calcula o valor do seno usando a série de MacLaurin. A função seno2 usa uma implementação normal python. Complete a função similar **seno**(x,nterms) usando funções NumPy. Recorra à função *np.vectorize*() para definir a função que será aplicada a todos os elementos de um array np (veja o exemplo no código fornecido).

$$f(x) = sen(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

3. Acrescente ao exercício 9.4 (Tabela de Futebol) o código para fazer um gráfico de barras da classificação, representando uma barra com os pontos de cada clube. Apresente o gráfico ordenado pela pontuação. Gere em compreensão duas listas, uma com os clubes e outra com os pontos, que serão os argumentos da função *plt.bar()* ou *plt.barh()*.

4. Uma imagem é um exemplo típico de uma matriz de 3 dimensões, em que a cada pixel do array de 2 dimensões, que definem o tamanho da imagem (largura e altura), está associado a 3ª dimensão que define a cor, que é representada por um array de 3 posições com os valores dos componentes que a formam (Red, Green, Blue). O programa imagem.py lê uma imagem jpeg para uma matriz NumPy, retira as componentes R e G de cada pixel, deixando só a componente azul, e grava a nova imagem. Analise o programa e a imagem resultante e depois acrescente as funcionalidades:

    a) Adicione o código para reduzir a imagem original 8 vezes, salvar a nova imagem, rodada de 90º (transposição da matriz nos axis 0 e axis 1, mantendo-se o axis 2) e mostrá-la. *Compare as imagens e o tamanho do ficheiro gerado com a imagem menor com o tamanho original e veja a enorme redução de espaço, correspondendo a uma menor qualidade da imagem.*

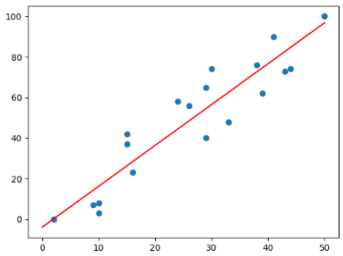
    b) Adicione o código para criar uma nova imagem em tons cinzentos e guardá-la usando o imsave(….., cmap = 'gray'). Uma imagem cinza tem 2 dimensões, só tem um valor para cada pixel, pelo que uma aproximação simples é fazer a média das 3 cores, usando o *np.mean*() só sobre o axis 2.

5. Acrescente ao programa *histograma.py* o código necessário para partindo do texto dado, o filtre criando uma lista só com letras, depois use a função histograma() que devolve um dicionário com a frequência das letras. De seguida deve fazer dois gráficos, lado a lado, um usando a função *plt.bar()* e outro usando a função *plt.hist()* da biblioteca MatPlotLib. *Note que a plt.hist() faz ela própria o histograma, sendo passados todos os dados originais*.

6. Regressão Linear: Como resultado do estudo de 20 alunos para um exame obteve-se um gráfico bidimensional (número de horas de estudo e nota de cada aluno) com um conjunto de pontos que, ao que tudo indica, parecem obedecer a uma relação do tipo y = mx + b.

   Pode-se implementar um programa que possa ser utilizado para obter os parâmetros m e b da recta de linearização dos pontos, atrás indicada, e que indique quão "boa" é a recta obtida relativamente à amostra, usando as expressões abaixo.
   Onde r é o Coeficiente de Correlação que mede o grau de associação linear existente entre os pontos (se |r|>0.9 o Grau de Associação Linear entre x e y é elevado, se |r|<0.1 o Grau de Associação Linear entre x e y é baixo).

$$m = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum (x_i^2) - (\sum x_i)^2},\quad b = \frac{\sum (x_i^2)\sum y_i - \sum x_i \sum x_i y_i}{n\sum (x_i^2) - (\sum x_i)^2} \quad \text{e} \quad r = \frac{\sum x_i y_i}{\sqrt{\sum (x_i^2)\sum (y_i^2)}}$$



   a) Complete o programa regressao.py de modo a fazer o gráfico scatered das notas versus horas, tal como indicado (a azul) na figura e de seguida deve calcular os coeficientes **m** e **b** que definem a recta que melhor se aproxima (a vermelho). Para isso use a função *polifit()* do NumPy como está descrita no ficheiro dado. Finalmente faça o gráfico da reta, a vermelho. *Compare esta solução NumPy com o que teria de fazer se tivesse de implementar as fórmulas dadas acima!*

   b) Infelizmente o NumPy não nos dá o valor de **r**! Calcule esse valor usando outras funções do NumPy.