Programação II

Exercícios 7
Gráficos de funções, de barras e histogramas com matplotlib

Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática
Licenciatura em Tecnologias da Informação

2020/2021

1. Vamos chamar *gráfico* a um par de listas de números com o mesmo comprimento. A primeira lista deve estar ordenada por ordem crescente e denota as *abcissas*, a segunda as *ordenadas*. Eis o gráfico da função quadrática tomada nos números inteiros entre 0 e 5:

   ```
   ([0, 1, 2, 3, 4, 5], [0, 1, 4, 9, 16, 25])
   ```

   Procuramos uma função `grafico` que, dada uma função de números em números, devolve o seu gráfico. A função `grafico` recebe quatro parâmetros, três dos quais opcionais. São eles: a função, a primeira abcissa `baixo = 0.0`, a última abcissa `alto = 10.0`, e o incremento entre abcissas `incremento = 1.0`. Exemplo:

   ```
   >>> grafico (lambda x: 2*x, alto = 5.0, incremento =
       1.7)
   ([0, 1.7, 3.4], [0, 3.4, 6.8])
   ```

   (a) Escreva uma solução utilizando listas de compreensão.

   (b) Escreva uma solução utilizando um ciclo **`for`**/**`while`**.

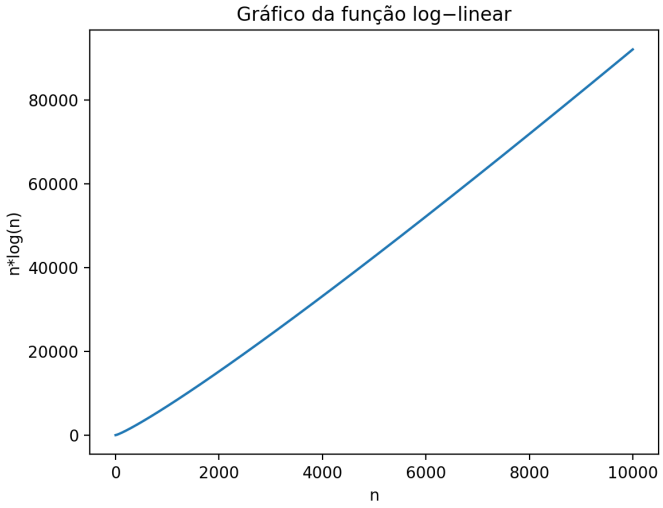
   (c) Escreva uma solução utilizando funções de ordem superior.

2. Escreva uma função `tracar_grafico` que trace o gráfico de uma função. A função deve receber um gráfico (um par de listas, ver exercício acima) e três parâmetros opcionais, a saber:
   `etiquetax = 'x'`,`etiquetay = 'f(x)'`,

`titulo = 'grafico da funcao f'`. Consulte o manual do Matplotlib, para descobrir as funções apropriadas.

Eis o aspeto da função $n \cdot \log(n)$, gerado pelo código abaixo.

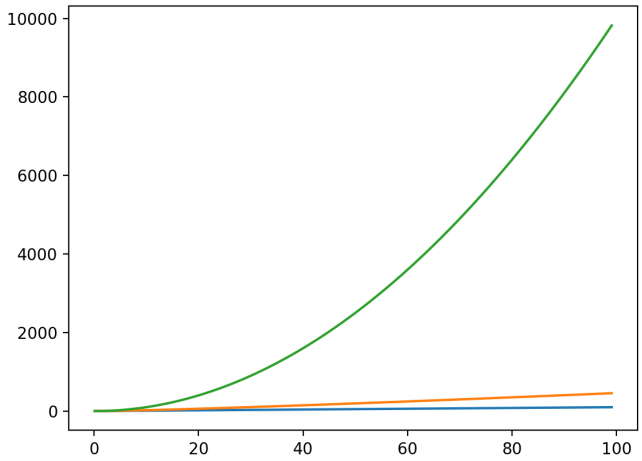
```
>>> import math
>>> g = grafico(lambda n: n*math.log(n), baixo = 1,
    alto = 10000)
>>> tracar_grafico(g, etiquetax = 'n', etiquetay = 'n
    *log(n)', titulo = 'Gráfico da função log-linear'
    )
```



3. Escreva uma função `tracar_graficos` que trace conjuntamente os gráficos de um número variável de funções. A função recebe como *único* parâmetro a lista dos vários gráficos (cada gráfico é um par de listas, ver exercício 1). Utilize uma função de ordem superior para iterar sobre os gráficos na lista. Por exemplo:

```
>>> baixo = 0.1
>>> alto = 100.0
>>> linear = grafico (lambda n: n, baixo = baixo,
    alto = alto)
>>> loglinear = grafico (lambda n: n*math.log(n),
    baixo = baixo, alto = alto)
>>> quadratico = grafico (lambda n: n**2, baixo =
    baixo, alto = alto)
>>> tracar_graficos ([linear, loglinear, quadratico])
```

deverá produzir um gráfico deste tipo:



4. (a) Escreva uma função `potencias(k)` que devolva gráficos (pares de listas) para as várias potências de um número. Dado um número k, a função deverá devolver uma lista com $k+1$ gráficos correspondentes às funções $x^0, x^1, x^2, \ldots, x^k$. Para além deste parâmetro (que é obrigatório), a função deverá receber três parâmetros opcionais, tal como no exercício 1. Utilize funções de ordem superior sempre que possível.

   (b) Utilizando a função da alínea anterior, bem como a função `tracar_graficos` do exercício 3, escreva uma função `tracar_potencias(k)` que trace os graficos das primeiras $k+1$ funções potência.

5. Escreva uma função `graficos` que devolva gráficos (pares de listas) para várias funções. A função recebe uma lista de funções e devolve uma lista de gráficos, um gráfico por função. Para além deste parâmetro (que é obrigatório), a função deverá receber três parâmetros opcionais, tal como no exercício 1. Utilize funções de ordem superior sempre que possível.

6. Podemos traçar um gráfico num par de eixos (caso do exercício 2), vários gráficos no mesmo par de eixos (caso do exercício 3) ou gráficos em diferentes pares de eixos (caso de este exercício). Todos os comandos `pyplot` dizem respeito ao par de eixos (abcissas, ordenadas) corrente. Os vários eixos estão organizados em linhas e colunas. Para escolher um novo par de eixos usamos a função

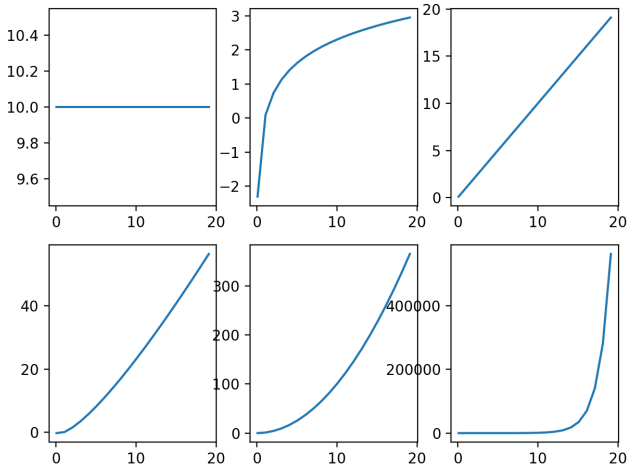
   ```
   pyplot.subplot(numero_de_linhas, numero_de_colunas,
       numero_dos_eixos)
   ```

onde o número dos eixos deverá estar entre 1 e o produto das linhas pelas colunas.

(a) Escreva uma função `tracar_subgrafico` que dado um gráfico (isto é, um par de listas ordenadas-abcissas), o número de linhas e de colunas, e o número do gráfico, construa um novo gráfico (`pyplot.plot`) num dado par de eixos.

(b) Escreva uma função `tracar_subgraficos` que dado uma lista de gráficos e o número de linhas construa uma figura com tantos gráficos quantos os presentes na lista. Por exemplo, o código Python abaixo deve imprimir o gráfico na figura também abaixo.

```
>>> baixo = 0.1
>>> alto = 20.0
>>> constante = grafico(lambda x: 10.0, baixo=
    baixo, alto=alto)
>>> logaritmico = grafico(lambda x: math.log(x),
    baixo=baixo, alto=alto)
>>> linear = grafico(lambda x: x, baixo=baixo,
    alto=alto)
>>> loglinear = grafico(lambda x: x*math.log(x),
    baixo=baixo, alto=alto)
>>> quadratico = grafico(lambda x: x*x, baixo=
    baixo, alto=alto)
>>> exponencial = grafico(lambda x: 2**x, baixo=
    baixo, alto=alto)
>>> tracar_subgraficos([constante, logaritmico,
    linear, loglinear, quadratico, exponencial],2)
```



(c) Escreva uma variante `tracar_subgraficos_sqrt` da função da alínea anterior que recebe apenas uma lista de gráficos. Esta função deverá escolher automaticamente o número 'ideal' de linhas de modo a que a grelha de gráficos fique o mais quadrada possível.

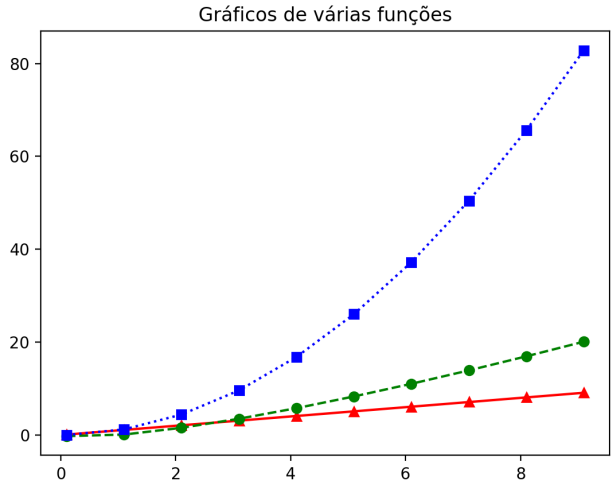
7. De modo a distinguir melhor os diferentes gráficos a traçar num par de eixos podemos personalizar a sua visualização. Assim, é possível adicionar a cada *plot* uma string de formatação. Por exemplo,

```
pyplot.plot(abcissas, ordenadas, 'go--')
```

faz com que o gráfico seja traçado em em cor verde (`'g'` de *green*), com marcas circulares (`'o'`) e linha tracejada (`'--'`). Todas a opções de formatação podem ser consultadas na documentação da função `plot`.

Escreva uma função `tracar_graficos_personalizados` que recebe dois argumentos: uma lista de gráficos e uma lista de strings de formatação com as formatações pretendidas para cada um dos gráficos fornecidos. As duas listas deverão ter o mesmo comprimento. A função deverá traçar os gráficos dados de acordo com as respectivas strings de formatação. Por exemplo, o código Python abaixo deve imprimir o gráfico na figura também abaixo.

```
>>> baixo = 0.1
>>> alto = 10
>>> linear = grafico(lambda n: n, baixo=baixo, alto=
    alto)
>>> loglinear = grafico(lambda n: n * math.log(n),
    baixo=baixo, alto=alto)
>>> quadratico = grafico(lambda n: n**2, baixo=baixo,
     alto=alto)
>>> formatacoes = ['r^-','go--','bs:']
>>> tracar_graficos_personalizados ([linear,
    loglinear, quadratico], formatacoes)
```



8. Escreva uma função `maximos` que, dada uma lista de gráficos, devolve uma lista com os máximos de cada um dos gráficos. A lista a devolver deverá ser uma lista de pares (abcissa, ordenada) correspondentes aos pontos máximos de cada gráfico.

9. (a) Escreva uma função `grafico_media` que, dada uma lista de gráficos, constrói um novo gráfico no qual, para cada abcissa, a ordenada é a média das ordenadas dos vários gráficos fornecidos. Assuma que todos os gráficos têm a mesma lista de abcissas.

   (b) Utilizando a função `grafico_media`, definida na alínea anterior, defina uma função `tracar_com_media` que recebe uma lista de gráficos e traça, num mesmo sistema de eixos, todos esse gráficos e ainda o gráfico das médias, sendo que este deverá ser apresentado com marcas circulares. Sugestão: recorra também à função `tracar_graficos_personalizados` do exercício 7.

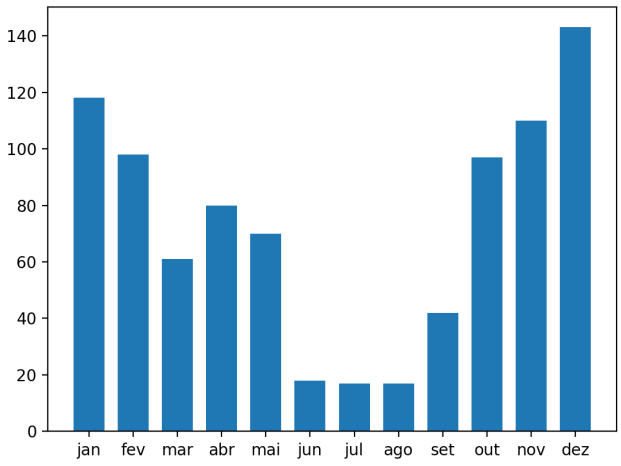
10. (a) Escreva uma função `grafico_maximo` que, dada uma lista de gráficos, constrói um novo gráfico no qual, para cada abcissa, a ordenada é o máximo das ordenadas dos vários gráficos fornecidos. Assuma que todos os gráficos têm a mesma lista de abcissas.

    (b) Utilizando a função `grafico_maximo`, definida na alínea anterior, defina uma função `tracar_com_maximo` que recebe uma lista de gráficos e traça, num mesmo sistema de eixos, todos esse gráficos e ainda o gráfico dos máximos, sendo que este deverá ser apresentado com marcas circulares. Sugestão: recorra também à função `tracar_graficos_personalizados` do exercício 7.

11. Escreva uma função booleana `e_maxima` que, dados dois gráficos, verifica se o primeiro é maior ou igual que o segundo em todos os pontos fornecidos, devolvendo, nesse caso, o valor `True`. Deverá devolver `False` caso exista pelo menos uma abcissa para a qual as ordenadas não verifiquem a condição indicada.

12. Defina uma função `grafico_barras` que, dada um dicionário, apresente um gráfico de barras no qual os nomes das barras correspondem às chaves do dicionário e as alturas das barras correspondem aos respetivos valores. Defina a função de modo a que, para o dicionário abaixo seja apresentado um gráfico do tipo também abaixo:

```
precipitacao = {'jan': 118, 'fev': 98, 'mar': 61, '
    abr': 80, 'mai': 70, 'jun': 18, 'jul': 17, 'ago':
     17, 'set': 42, 'out': 97, 'nov': 110, 'dez':
    143}
```



Sugestões. Comece por programar a função `unzip` que, dada um dicionário, devolve um par de listas com as chaves e os valores respetivamente. Por exemplo:

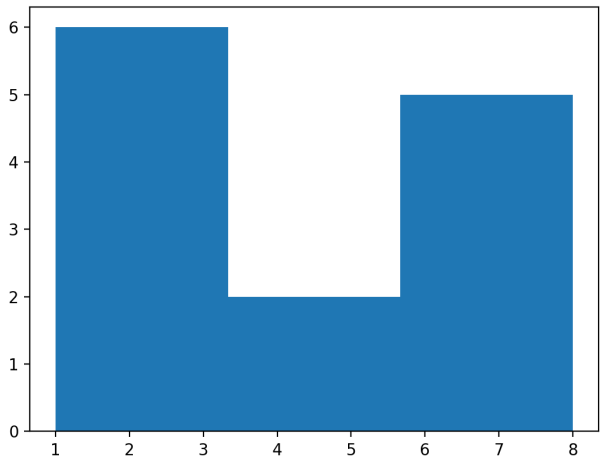
```
>>> unzip ({'jan': 118, 'fev': 98, 'mar': 61})
(['jan', 'fev', 'mar'], [118, 98, 61])
```

Utilize a função `pyplot.xticks(posição_das_etiquetas, etiquetas)`. Os

parâmetros são duas listas que devem ter o mesmo comprimento. O primeiro parâmetro dá as posições em torno das quais o texto da etiqueta é centrado.

13. A função `pyplot.hist(valores,num_classes)` permite representar um conjunto de valores sob a forma de um histograma com o número de classes dado. Por exemplo, o código Python abaixo deve imprimir o gráfico na figura também abaixo:

```
>>> dados = [1, 1, 1, 3, 2, 5, 1, 5, 6, 6, 7, 8, 8]
>>> pyplot.hist(dados, 3)
```

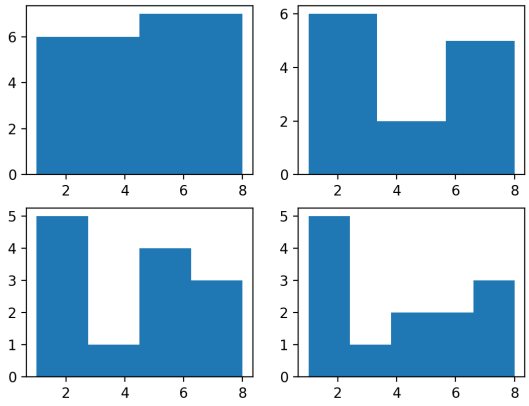


Defina uma função `histogramas_classes` que permita mostrar várias representações em histograma de um mesmo conjunto de valores, considerando um número variável de classes. A função recebe dois argumentos: a lista de valores para construir o histograma e uma lista de inteiros. Cada inteiro na segunda lista representa um número de classes. A função apresenta tantos histogramas quantos os elementos desta segunda lista. Defina a função de modo a que:

(a) Cada histograma seja apresentado numa figura independente.

(b) Os histogramas apareçam todos numa mesma figura, havendo, no máximo, 2 por coluna. Utilize a função `pyplot.subplot` e as ideias do exercício 6. Por exemplo, o código seguinte deverá mostrar a figura apresentada.

```
>>> dados = [1, 1, 1, 3, 2, 5, 1, 5, 6, 6, 7, 8,
    8]
>>> histogramas_classes(dados, [2, 3, 4, 5])
```



14. Escreva uma função `traca_frequencias(nome_ficheiro)` que lê um ficheiro de texto e apresenta um gráfico de barras com as frequências de cada letra no texto. Para simplificar, considere apenas letras minúsculas `abc...xyz` e ignore pontuação e acentuação.

    Sugestões. Utilize um dicionário em que as chaves correspondem às vinte e seis letras do alfabeto, e os valores correspondem às respetivas frequências. Defina uma função `dicionario_frequencias` que leia o ficheiro de texto e devolva o respetivo dicionário de frequências. Utilize a função `grafico_barras` do exercício 12.

    Por exemplo, para o primeiro canto d'Os Lusíadas (ficheiro `lusiadas.txt` disponível no Moodle), temos a seguinte distribuição: