Politécnico de Leiria

Escola Superior de Tecnologia e Gestão

Matemática Discreta - Componente PL

EI (D)

Ano letivo 2023/2024 - 2.º Sem. Ficha prática 2 (PL)

Vetores

Em Python existem várias formas de criar vetores e matrizes e operar com os mesmos. Nesta UC, para definir e manipular vetores e matrizes, será usado o *package Numpy*. Sugere-se a importação deste *package*, utilizando a convenção `np`, para trabalhar com os seus objetos ao longo do código:

```
In [1]: import numpy as np
```

Os vetores e matrizes podem ser definidos através do objeto *array*, que pertence a este *package*. Para definir um vetor, pode ser utilizado um array unidimensional; para definir uma matriz (com mais de uma coluna ou mais de uma linha) será usado um array bidimensional.

- Um **vetor** pode ser criado de várias formas:

Comandos para a construção de um vetor	
`x=np.array([x0,x1,...,xn])`	Define o vetor `x` com $(n+1)$ elementos
`x=np.arange(x0,x1)`	Cria um vetor `x` cujo primeiro elemento é o `x0`, sendo os seguintes obtidos a partir deste por incremento de 1 unidade, enquanto o valor resultante for inferior a `x1`.
`x=np.arange(x0,x1,step)`	Cria um vetor `x` cujo primeiro elemento é o `x0`, sendo os seguintes obtidos a partir deste pela adição do `step`, enquanto o valor resultante for inferior a `x1`.
`x=np.linspace(x0,x1,n)`	Cria um vetor `x` com `n` elementos, igualmente espaçados, cujo primeiro elemento é o `x0` e o último elemento é o `x1`.

Exemplo:

Para definir o vetor $v = [1, 3, 5, 7, 9]$ podemos usar uma das seguintes alternativas:

```
In [25]: v=numpy.array([1,3,5,7,9])
         print(v)

         [1 3 5 7 9]
```

```
In [26]: v=numpy.arange(1,10,2)
         print(v)

         [1 3 5 7 9]
```

```
In [27]: v=numpy.linspace(1,9,5)
         print(v)

         [1. 3. 5. 7. 9.]
```

- Para obter informações sobre um **vetor** ou acrescentar elementos, são úteis os comandos:

Funções sobre arrays	
`A.size`	Devolve o n.º de elementos da matriz `A`
`A.shape`	Devolve a dimensão da matriz `A` (n.º de linhas × n.º de colunas)
`np.append(v,x)`	Acrescenta o elemento `x` ao vetor `v`

Funções

- Uma função é um bloco de código que só é executado quando é chamado.
  A sintaxe de uma função é definida por três partes: nome, parâmetros e corpo, o qual agrupa uma sequência de linhas que se pretende que sejam executadas quando a função é chamada:

```
def nome(parâmetros):
    corpo
```

Exemplo:

```
In [30]: def hello(nome):
          print("Olá",nome)

In [31]: hello("João")

         Olá João
```

No exemplo anterior, o nome da função é `"hello"`. Dentro dos parênteses é colocado o argumento `nome` (dados de entrada). A utilização dos dois pontos (:) indica que o código identado nas linhas abaixo constitui o corpo da função que está a ser criada. Na linguagem Python, a identação é um fator determinante; neste caso, o `corpo` da função deve estar avançada em relação à 1^a linha.

Também é possível definir funções com nenhum argumento ou com vários argumentos.

Exemplos:

```
In [32]: def Welcome():
          print("Seja bem vindo!")

In [33]: Welcome()

         Seja bem vindo!


In [34]: def soma(a,b):
          return a+b

In [35]: S=soma(8,5)
         S

Out[35]: 13
```

In [36]:
```
def Operador(a,b):
 return a+b,a-b
#----------------------#
Resultado=Operador(3,7)
print(Resultado)
print(Resultado[0])
print(Resultado[1])
```

```
(10, -4)
10
-4
```

In [37]:
```
def Operador(a,b):
 return a+b,a-b
#----------------------#
soma,sub=Operador(3,7)
print(soma)
print(sub)
```

```
10
-4
```

Visualização gráfica

- Para criar um gráfico de uma função de uma ou de duas variáveis real, é aconselhado usar a biblioteca *Mathplotlib*. Sugere-se a importação do package `matplotlib.pyplot` utilizando a convenção `plt`.

- Para fazer a representação gráfica de um conjunto de pontos, pode usar-se a função `plot`, contida neste *package*. Existe uma extensa lista de opções a considerar na construção de um gráfico (ver https://matplotlib.org/2.0.2/index.html).

- De seguida são apresentadas algumas das mais importantes instruções que permitem definir as características de uma representação gráfica em duas dimensões.

Visualização gráfica	
`plt.plot(X,Y)`	Representa a lista de pontos (X, Y) cujas abcissas são os elementos do vetor X e as ordenadas são os elementos do vetor Y
`plt.plot(X, Y,'S')`	A sequência de caracteres `'S'` permite definir características da representação (`'S'` é constituída por elementos das colunas da tabela seguinte)

Nota: Os vetores `X e Y` podem ser definidos usando qualquer um dos comandos de definição de arrays estudados anteriormente.

Caso tenha sido definida uma função `f`, o vector `Y` pode ser definido pelo comando `Y=f(X)`.

- A seguinte tabela apresenta os caracteres correspondentes às várias características possíves de definir para um determinado gráfico, através da string `'S'`:

Caractere	Cor	Caractere	Marcador	Caractere	Tipo de linha
`b`	azul	`.`	ponto	`-`	linha contínua
`g`	verde	`o`	círculo	`- -`	linha tracejada
`r`	vermelho	`x`	X	`-.`	traço e ponto
`c`	ciano	`+`	+	`:`	linha pontilhada
`m`	magenta	`*`	estrela		
`y`	amarelo	`s`	quadrado		
`k`	preto	`d`	diamante		
`w`	branco	`v`	triângulo para baixo		

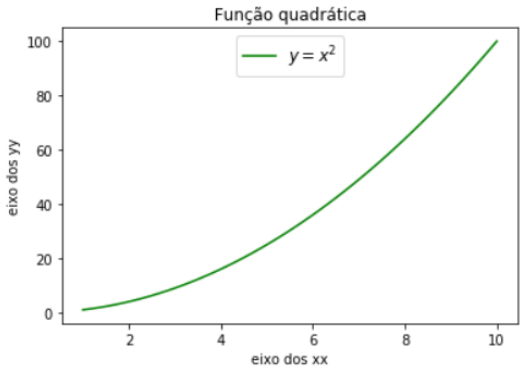
- É possível também personalizar os gráficos com diversas opções relacionadas com o controle dos eixos e com a inserção de texto. Na tabela seguinte são apresentadas algumas instruções para inserir título e legendas nos gráficos (para saber mais, consultar https://www.w3schools.com/python/matplotlib_labels.asp).

Inserção de texto em gráficos	
`plt.legend(...)`	Legenda das curvas do gráfico
`plt.title(string_título)`	Título do gráfico
`plt.xlabel(string_legenda_xx)`	Legenda do eixo dos xx
`plt.ylabel(string_legenda_yy)`	Legenda do eixo dos yy

Exemplo:

In [38]:
```
import matplotlib.pyplot as plt
def f(x):
 return x**2

xdata = np.linspace(1,10,50)
ydata = f(xdata)
plt.plot (xdata,ydata,"g")
plt.title(" Função quadrática ")
plt.xlabel("eixo dos xx")
plt.ylabel("eixo dos yy")
plt.legend(["$y=x^2$"],loc='upper center',fontsize='12')
plt.show()
```



Estruturas de controlo

O Python contém diversas estruturas que permitem controlar a sequência do código executado de acordo com dados iniciais ou com dados obtidos ao longo do mesmo, como por exemplo a estrutura de seleção `if` ou as estruturas de repetição `for` e `while`.

- Estrutura de seleção **if**

A instrução `if` avalia uma proposição e executa um grupo de instruções se essa proposição for verdadeira. Se a proposição tiver valor lógico falso, as instruções a executar podem opcionalmente ser definidas através do comando `else` ou `elif`, de acordo com a seguinte sintaxe:

```
if condição lógica 1:
    instruções executáveis se a condição lógica 1 for verdadeira
elif condição lógica 2:
    instruções executáveis se a condição lógica 2 for verdadeira (e se cond.lógica 1 falsa)
else
    instruções executáveis a realizar quando não são verificadas as condições 1 e 2
```

Exemplos:

In [24]:
```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

```
a is greater than b
```

In [25]:
```
a = 66/2
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

```
a and b are equal
```

- Estrutura de repetição **for**

A estrutura de repetição `for` executa um conjunto de instruções para cada item pertencente a uma lista, tupla, conjunto, entre outros.
A sintaxe do ciclo `for` é:

```
for índice in sequência:
    instruções executáveis
```

Exemplos:

```
In [26]: for x in range(2, 30, 3):
           print(x)
```

```
2
5
8
11
14
17
20
23
26
29
```

A declaração `break`:

Com a instrução `break` é possível parar a estrutura de repetição antes que esta termine:

Exemplo:

```
In [27]: for x in range(2, 30, 3):
          print(x)
          if x==17:
           break
```

```
2
5
8
11
14
17
```

- Estrutura de repetição **while**

Nesta estrutura de repetição as instruções especificadas são executadas enquanto a condição for verdadeira, seguindo a sintaxe:

```
while condição lógica:
    instruções executáveis
```

Exemplo:

```
In [29]: i = 1
         while i < 6:
          print(i)
          if i == 3:
           break
          i += 1
```

```
1
2
3
```

Exercícios

1. Seja f a função definida por $f(x) = \cos(x) + e^x$.
   (a) Defina/implemente a função f.
   (b) Determine o valor de $f(0)$ e de $f(\pi)$.
   (c) Construa um vetor linha X cujo primeiro elemento seja `1`, o último seja `2` e tenha um total de 98 elementos.
   (d) Calcule $f(X)$. O que acontece?
   (e) Faça a representação gráfica da função f.

2. Faça a representação gráfica da função polinomial definida por

$$f(x) = x^5 - 3x^4 - 3x^3 + 7x^2 + 6x$$

   no intervalo $[-1,5; 2,5]$ com incremento de $0,125$.

3. O polónio tem uma meia-vida de 140 dias, o que significa que, devido ao decaimento radioactivo, a quantidade de polónio restante depois de 140 dias é metade da original. A quantidade restante, após um certo período de tempo t, é dada por

$$r(t) = C_0(0.5)^{t/v}, \quad \text{(t em dias)}$$

   sendo C_0 a quantidade inicial e v o tempo de meia-vida. Se hoje tivermos 10 gramas de polónio, qual a quantidade restante ao fim de cada uma das próximas 10 semanas? Elabore um gráfico que mostre o comportamento observado ao longo desse período, desde o instante inicial.

4. Considere a sucessão Z definida por $Z(n) = \sum_{k=1}^{n} \left(\frac{1}{2}\right)^k$.

   (a) Faça uma representação gráfica de Z onde se possam visualizar os seus 20 primeiros termos.

   (b) Determine o primeiro valor de n tal que seja verificada a condição $|Z(n) - Z(n-1)| < 10^{-10}$.

5. Considere a sucessão S de termo geral

$$S_n = \begin{cases} 2^{-\frac{n}{10}} \times \frac{n}{3} & se \quad n \text{ é múltiplo de 3} \\ 3^{-\frac{n}{10}} \times n & se \quad n \text{ não é múltiplo de 3} \end{cases}, \qquad n \geq 1.$$

   (a) Represente graficamente os 70 primeiros termos da sucessão S.

   (b) Construa uma função `soma(n)` que receba um inteiro `n` e devolva a soma dos primeiros `n` termos da sucessão S_n; utilize-a para determinar o primeiro valor de `n` tal que a soma dos `n` primeiros termos é superior a 42.