

Programando em Python

Aula 5

Prof. Dr. Marco Antonio Leonel Caetano

Criando *Vetores (array)*

Criando *Vetores (array)*

Variáveis indexadas - VETORES

- **O que são?**

- variáveis com índices para armazenar n-valores

- **Utilidade?**

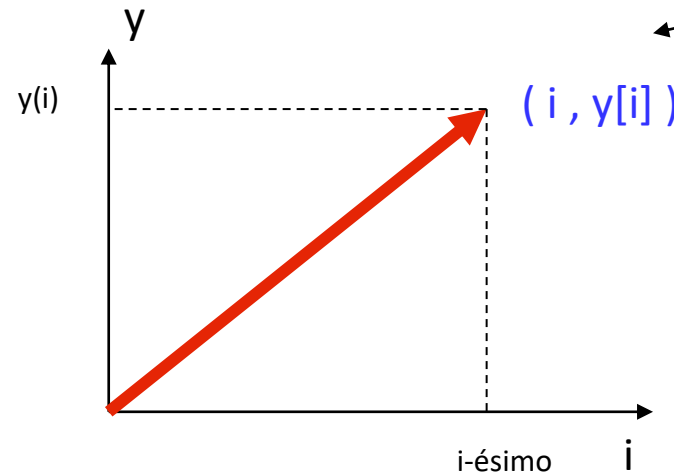
- representação espacial de soluções.
- aumento de rapidez de processamento.
- melhorar a estruturação de um programa.
- estudos e análises de estabilidades.

- **A notação em computação (Python)**

- é necessário um índice inteiro “i”, “j”, ...
- A representação segue-se um colchete depois do nome da variável: y[i], z[k], w[j], etc.

Criando *Vetores (array)*

A representação espacial



Par ordenado:
Abscissa: i
Ordenada: $y[i]$

Criando *Vetores (array)*

A representação algébrica de vetores

$$\vec{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

Ou seja,

$$\vec{y} = \begin{pmatrix} y[1] \\ y[2] \\ \vdots \\ y[n] \end{pmatrix}$$

→ y para o 1º valor de y

→ y para o 2º valor de y

→ y para o último valor de y

Criando *Vetores (array)*

Para ler os valores e preencher um vetor “v”

```
1 import numpy as np
2
3 n=int(input("número de termos = "))
4 v=np.zeros(n)
```



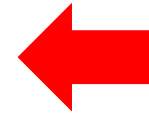
Ao contrário de outras linguagens, no Python um vetor deve ser iniciado por zero ou tomar elementos de listas já preenchidas

V = [0 0 0 0 0 0 0 ... 0]

Criando *Vetores (array)*

Para ler os valores e preencher um vetor “v”

```
1 import numpy as np
2
3 n=int(input("número de termos = "))
4 v=np.zeros(n)
5 for i in range(n):
6     v[i]=float(input("valor= "))
```



Lendo um a um os elementos

Criando *Vetores (array)*

Para ler os valores e preencher um vetor “v”

```
1 import numpy as np
2
3 n=int(input("número de termos = "))
4 v=np.zeros(n)
5 for i in range(n):
6     v[i]=float(input("valor= "))
7
8 print(v)
```

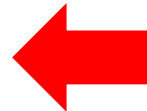


Imprime todo o vetor

Criando *Vetores (array)*

Para ler os valores e preencher um vetor “v”

```
1 import numpy as np
2
3 n=int(input("número de termos = "))
4 v=np.zeros(n)
5 for i in range(n):
6     v[i]=float(input("valor= "))
7
8 print(v)
```



Imprime todo o vetor

Resultado

número de termos = 3

valor= -1.4

valor= 2.5

valor= -10.2

[-1.4 2.5 -10.2]

Criando *Vetores (array)*

x	y(x)
1	y(1)=3
1.1	y(2)=3.3
1.2	y(3)=3.6
1.3	y(4)=4.2

- Apesar do espaçamento do x ser de 0.1, a representação computacional em y continua tendo como referência a ordem dos valores de x e **NÃO** os próprios valores de x.
- Os **ÍNDICES** de y(x) serão sempre números inteiros demarcando as posições dos valores de x.

Criando *Vetores (array)*

x	y(x)
1	y(1)=3
1.1	y(2)=3.3
1.2	y(3)=3.6
1.3	y(4)=4.2

```
1 import numpy as np
2
3 n=int(input("número de termos = "))
4 x=np.zeros(n)
5 y=np.zeros(n)
6 for i in range(n):
7     x[i]=float(input("x = "))
8     y[i]=float(input("y[x]= "))
9
10 print(x,y)
```

Criando *Vetores (array)*

x	y(x)
1	y(1)=3
1.1	y(2)=3.3
1.2	y(3)=3.6
1.3	y(4)=4.2

```
1 import numpy as np
2
3 n=int(input("número de termos = "))
4 x=np.zeros(n)
5 y=np.zeros(n)
6 for i in range(n):
7     x[i]=float(input("x = "))
8     y[i]=float(input("y[x]= "))
9
10 print(x,y)
```

Resultado

```
número de termos = 4

x = 1
y[x]= 3
x = 1.1
y[x]= 3.3
x = 1.2
y[x]= 3.6
x = 1.3
y[x]= 4.2
[1.  1.1 1.2 1.3] [3.  3.3 3.6 4.2]
```

Operações com *Vetores (array)*

x	y(x)=3x
1	y(1)=3
1.1	y(2)=3.3
1.2	y(3)=3.6
1.3	y(4)=4.2

```
1 import numpy as np
2
3 n=int(input("número de termos = "))
4 x=np.zeros(n)
5 y=np.zeros(n)
6 for i in range(n):
7     x[i]=float(input("x = "))
8     y[i]=float(input("y[x]= "))
9
10 print(x,y)
```

Resultado

```
número de termos = 4

x = 1
y[x]= 3
x = 1.1
y[x]= 3.3
x = 1.2
y[x]= 3.6
x = 1.3
y[x]= 4.2
[1.  1.1 1.2 1.3] [3.  3.3 3.6 4.2]
```

Operações com *Vetores (array)*


x	y(x)=3x
1	y(1)=3
1.1	y(2)=3.3
1.2	y(3)=3.6
1.3	y(4)=4.2

```
1 import numpy as np
2
3 n=int(input("número de termos = "))
4 x=np.zeros(n)
5 y=np.zeros(n)
6 for i in range(n):
7     x[i]=float(input("x = "))
8     y[i]=3*x[i]
9
10 print(x,y)
```

Operações com *Vetores (array)*

x	y(x)=3x
1	y(1)=3
1.1	y(2)=3.3
1.2	y(3)=3.6
1.3	y(4)=4.2

```
1 import numpy as np
2
3 n=int(input("número de termos = "))
4 x=np.zeros(n)
5 y=np.zeros(n)
6 for i in range(n):
7     x[i]=float(input("x = "))
8     y[i]=3*x[i]
9
10 print(x,y)
```



y[i] é calculado agora

Resultado

número de termos = 4

x = 1

x = 1.1

x = 1.2

x = 1.3

[1. 1.1 1.2 1.3] [3. 3.3 3.6 3.9]

Operações com *Vetores (array)*

Uma vez conhecidos ou criados os vetores, pode-se realizar operações algébricas como se fossem variáveis simples.

Exemplo:

Ler os dois vetores x e y

$$\vec{x} = \begin{pmatrix} 2 \\ -1 \\ 4 \\ 3 \end{pmatrix} \quad \vec{y} = \begin{pmatrix} 5 \\ 3 \\ 10 \\ -2 \end{pmatrix}$$

Operações com *Vetores (array)*

Imprimir x+y

```
1 import numpy as np
2 import matplotlib.pyplot as fig
3 from math import cos, exp
4
5 n=int(input("número de termos = "))
6 x=np.zeros(n)
7 y=np.zeros(n)
8 for i in range(n):
9     x[i]=float(input("x= "))
10    y[i]=float(input("y= "))
11
```

Leitura

Operações com *Vetores (array)*

Imprimir x+y

```
1 import numpy as np
2 import matplotlib.pyplot as fig
3 from math import cos, exp
4
5 n=int(input("número de termos = "))
6 x=np.zeros(n)
7 y=np.zeros(n)
8 for i in range(n):
9     x[i]=float(input("x= "))
10    y[i]=float(input("y= "))
11
12 soma=x+y
13 print("++++++ soma +++++")
14 print(soma)
```



A soma não precisa de “for”

Operações com *Vetores (array)*

```
número de termos = 4  
  
x= 2  
  
y= 5  
  
x= -1  
  
y= 3  
  
x= 4  
  
y= 10  
  
x= 3  
  
y= -2  
++++++ soma ++++  
[ 7.  2. 14.  1.]
```

Operações com *Vetores (array)*

E continuando no console de saídas:

$x - y$ In [27]: `print(x-y)`
 [-3. -4. -6. 5.]

Operações com *Vetores (array)*

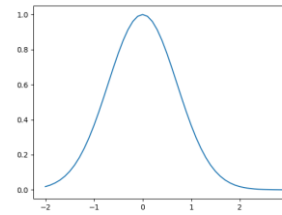
E continuando no console de saídas:

$x - y$ In [27]: `print(x-y)`
 [-3. -4. -6. 5.]

x / y (divide elemento a elemento de cada vetor)

In [29]: `print(x/y)`
[0.4 -0.33333333 0.4 -1.5]

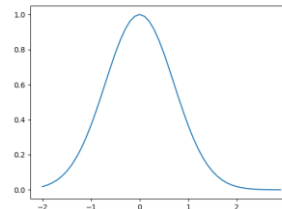
Estatísticas com *Vetores (array)*



Somar todos os elementos de x (não precisa de “*for*”). Basta usar o (.) como classe

```
In [30]: x.sum()  
Out[30]: 8.0
```

Estatísticas com *Vetores (array)*



Somar todos os elementos de x (não precisa de “*for*”). Basta usar o (.) como classe

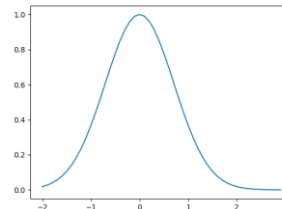
```
In [30]: x.sum()  
Out[30]: 8.0
```

Média de x e y

```
In [31]: x.mean()  
Out[31]: 2.0
```

```
In [32]: y.mean()  
Out[32]: 4.0
```

Estatísticas com *Vetores (array)*



Somar todos os elementos de x (não precisa de “*for*”). Basta usar o (.) como classe

```
In [30]: x.sum()  
Out[30]: 8.0
```

Média de x e y

```
In [31]: x.mean()  
Out[31]: 2.0
```

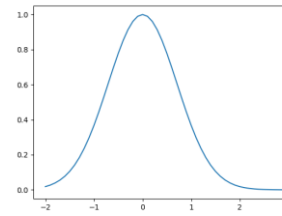
```
In [32]: y.mean()  
Out[32]: 4.0
```

Desvio Padrão de x e y

```
In [33]: x.std()  
Out[33]: 1.8708286933869707
```

```
In [34]: y.std()  
Out[34]: 4.301162633521313
```

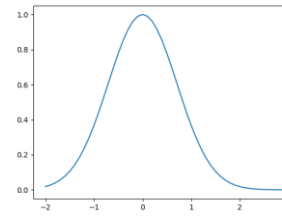

Estatísticas com *Vetores (array)*



Menor valor de x

```
In [35]: x.min()  
Out[35]: -1.0
```

Estatísticas com *Vetores (array)*



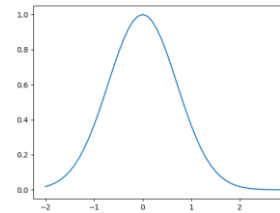
Menor valor de x

```
In [35]: x.min()  
Out[35]: -1.0
```

Maior valor de x

```
In [36]: x.max()  
Out[36]: 4.0
```

Estatísticas com *Vetores (array)*



Menor valor de x

```
In [35]: x.min()  
Out[35]: -1.0
```

Maior valor de x

```
In [36]: x.max()  
Out[36]: 4.0
```

Índice do menor x

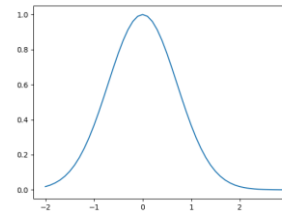
```
In [37]: x.argmin()  
Out[37]: 1
```

$$\vec{x} = \begin{pmatrix} 2 \\ -1 \\ 4 \\ 3 \end{pmatrix}$$

Índices do vetor

[0]
[1]
[2]
[3]

Estatísticas com *Vetores (array)*



Índice do maior x

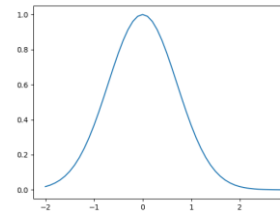
```
In [38]: x.argmax()  
Out[38]: 2
```

$$\vec{x} = \begin{pmatrix} 2 \\ -1 \\ 4 \\ 3 \end{pmatrix}$$

Índices do vetor

[0]
[1]
[2]
[3]

Estatísticas com *Vetores (array)*



Índice do maior x

```
In [38]: x.argmax()  
Out[38]: 2
```

Soma acumulada de x

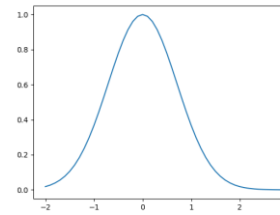
```
In [39]: x.cumsum()  
Out[39]: array([2., 1., 5., 8.])
```

$$\vec{x} = \begin{pmatrix} 2 \\ -1 \\ 4 \\ 3 \end{pmatrix}$$

Índices do vetor

[0]
[1]
[2]
[3]

Estatísticas com *Vetores (array)*



Índice do maior x

```
In [38]: x.argmax()  
Out[38]: 2
```

Soma acumulada de x

```
In [39]: x.cumsum()  
Out[39]: array([2., 1., 5., 8.])
```

Produto acumulado de x

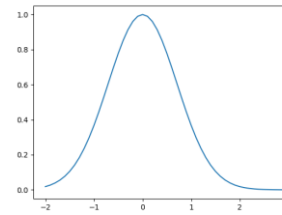
```
In [40]: x.cumprod()  
Out[40]: array([ 2., -2., -8., -24.])
```

$$\vec{x} = \begin{pmatrix} 2 \\ -1 \\ 4 \\ 3 \end{pmatrix}$$

Índices do vetor

[0]
[1]
[2]
[3]

Estatísticas com *Vetores (array)*



- **sum()** : computa a soma de todos os elementos em um array ou de um de seus eixos;
- **mean()** : computa média de todos os elementos em um array ou de seus eixos (células com valor NaN são ignoradas);
- **var()** e **std()** : variância e desvio padrão, com graus de liberdade podendo ser ajustados via parâmetro (o valor *default* é n);
- **min()** e **max(a)** : menor e maior valor de um array ou de um de seus eixos;
- **argmin()** e **argmax()** : índice do menor e maior valor;
- **cumsum()** : soma cumulativa dos elementos, começando pelo valor 0;
- **cumprod()** : produto acumulado dos elementos, começando pelo valor 1;

Gráfico com *Vetores (array)*

Exemplo: Fazer o gráfico de $y(x) = e^{-x} \cos(x)$, $x > 0$

```
1 import numpy as np
2 import matplotlib.pyplot as fig
3 import math
4
```



Importar a biblioteca gráfica

Gráfico com *Vetores (array)*

Exemplo: Fazer o gráfico de $y(x) = e^{-x} \cos(x)$, $x > 0$

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import math
4
```



Importar a biblioteca de matemática

Gráfico com *Vetores (array)*

Exemplo: Fazer o gráfico de $y(x) = e^{-x} \cos(x)$, $x > 0$

```
1 import numpy as np
2 import matplotlib.pyplot as fig
3 import math
4
5 n=int(input("número de termos = "))
6 x=np.zeros(n)
7 y=np.zeros(n)
8 for i in range(n):
9     x[i]=i*0.1
```

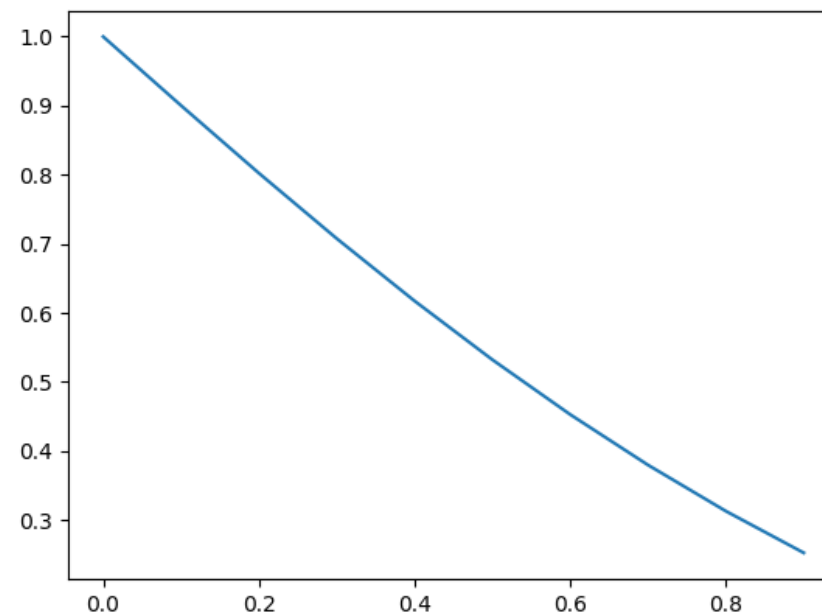


Criar o eixo x com pequeno espaçamento

Gráfico com *Vetores (array)*

Exemplo: Fazer o gráfico de $y(x) = e^{-x} \cos(x)$, $x > 0$

```
1 import numpy as np
2 import matplotlib.pyplot as fig
3 import math
4
5 n=int(input("número de termos = "))
6 x=np.zeros(n)
7 y=np.zeros(n)
8 for i in range(n):
9     x[i]=i*0.1
10    y[i]=math.cos(x[i])*math.exp(-x[i])
11
12 fig.plot(x,y)
```



Importando bibliotecas específicas do *Math*

*Muitas vezes não precisamos importar toda a biblioteca,
Mas apenas algumas funções específicas. Nesse caso, basta
Importar usando “**From**”.*

Exemplo: importar apenas cosseno e exponencial

```
from math import cos, exp
```

No exemplo anterior ...

```
1 import numpy as np
2 import matplotlib.pyplot as fig
3 from math import cos, exp
4
5 n=int(input("número de termos = "))
6 x=np.zeros(n)
7 y=np.zeros(n)
8 for i in range(n):
9     x[i]=i*0.1
10    y[i]=cos(x[i])*exp(-x[i])
11
12 fig.plot(x,y)
```


Não precisa usar o comando

math.<>

Eixo das abscissas negativo

Exemplo: Fazer o gráfico de $y(x) = e^{-x^2}$, $x > -2$

```
1 import numpy as np
2 import matplotlib.pyplot as fig
3 from math import cos, exp
4
5 n=int(input("número de termos = "))
6 x=np.zeros(n)
7 y=np.zeros(n)
8 for i in range(n):
9     x[i]=-2+i*0.1
10    y[i]=exp(-x[i]**2)
11
12 fig.plot(x,y)
```

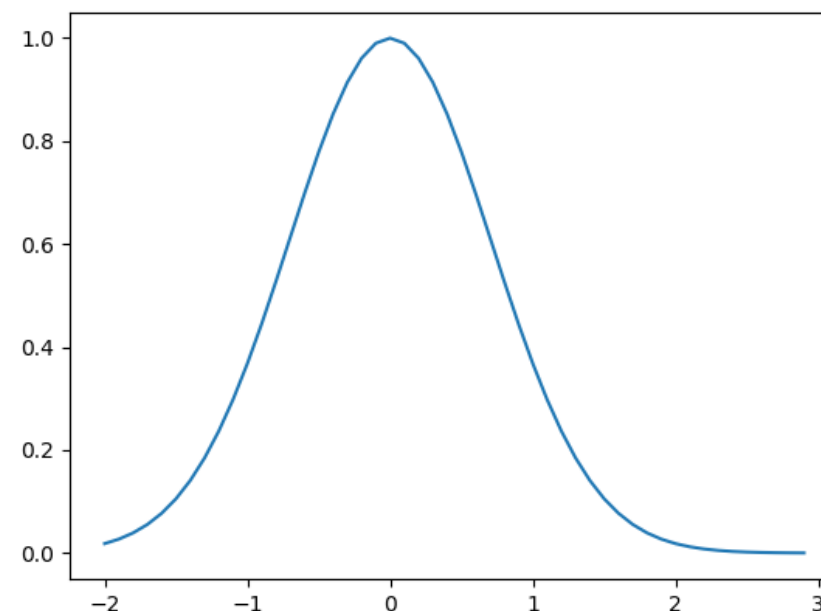
 Começa em x > -2

Eixo das abscissas negativo

Exemplo: Fazer o gráfico de $y(x) = e^{-x^2}$, $x > -2$

```
1 import numpy as np
2 import matplotlib.pyplot as fig
3 from math import cos, exp
4
5 n=int(input("número de termos = "))
6 x=np.zeros(n)
7 y=np.zeros(n)
8 for i in range(n):
9     x[i]=-2+i*0.1
10    y[i]=exp(-x[i]**2)
11
12 fig.plot(x,y)
```

N = 50 termos



Gráficos de diversos vetores (cores diferentes)

Exemplo: Fazer o gráfico de $y(x) = e^{-x^2}$, $z(x) = \cos(x)$ $x > -2$

Gráficos com diversos vetores (cores diferentes)

Exemplo: Fazer o gráfico de $y(x) = e^{-x^2}$, $z(x) = \cos(x)$ $x > -2$

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from math import cos, exp
4
5 n=int(input("número de termos = "))
6 x=np.zeros(n)
7 y=np.zeros(n)
8 z=np.zeros(n)
9 for i in range(n):
10     x[i]=-2+i*0.1
11     y[i]=exp(-x[i]**2)
12     z[i]=cos(x[i])
```

Gráficos de diversos vetores (cores diferentes)

Exemplo: Fazer o gráfico de $y(x) = e^{-x^2}$, $z(x) = \cos(x)$ $x > -2$

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from math import cos, exp
4
5 n=int(input("número de termos = "))
6 x=np.zeros(n)
7 y=np.zeros(n)
8 z=np.zeros(n)
9 for i in range(n):
10     x[i]=-2+i*0.1
11     y[i]=exp(-x[i]**2)
12     z[i]=cos(x[i])
13 fig.plot(x,y,'-b',x,z,'-r')
```



Gráficos de diversos vetores (cores diferentes)

Exemplo: Fazer o gráfico de $y(x) = e^{-x^2}$, $z(x) = \cos(x)$ $x > -2$

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from math import cos, exp
4
5 n=int(input("número de termos = "))
6 x=np.zeros(n)
7 y=np.zeros(n)
8 z=np.zeros(n)
9 for i in range(n):
10     x[i]=-2+i*0.1
11     y[i]=exp(-x[i]**2)
12     z[i]=cos(x[i])
13 fig.plot(x,y,'-b',x,z,'-r')
```

VERMELHO



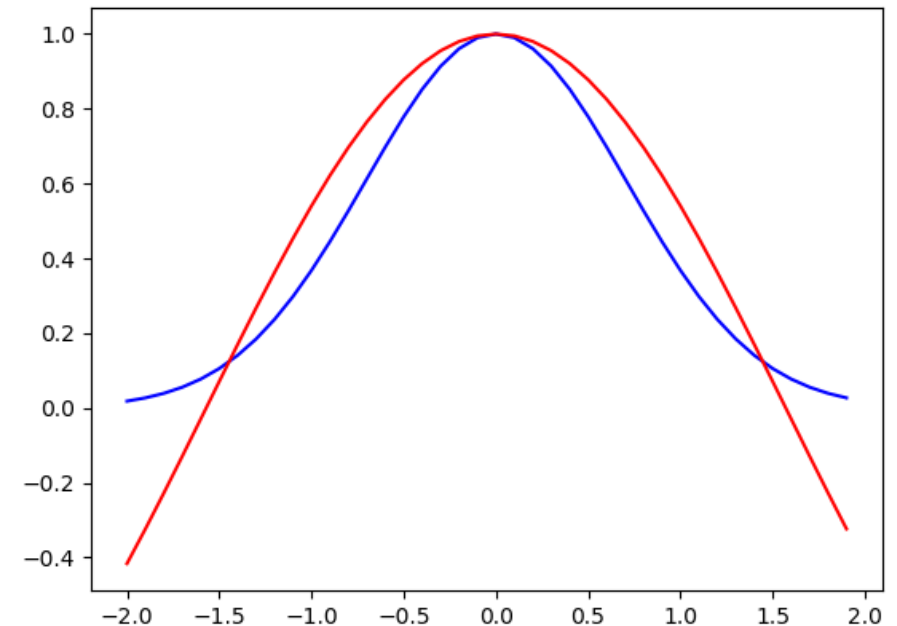
Gráficos de diversos vetores (cores diferentes)

Exemplo: Fazer o gráfico de $y(x) = e^{-x^2}$, $z(x) = \cos(x)$ $x > -2$

N = 40 termos

```
1 import numpy as np
2 import matplotlib.pyplot as fig
3 from math import cos, exp
4
5 n=int(input("número de termos = "))
6 x=np.zeros(n)
7 y=np.zeros(n)
8 z=np.zeros(n)
9 for i in range(n):
10     x[i]=-2+i*0.1
11     y[i]=exp(-x[i]**2)
12     z[i]=cos(x[i])
13 fig.plot(x,y,'-b',x,z,'-r')
```

VERMELHO

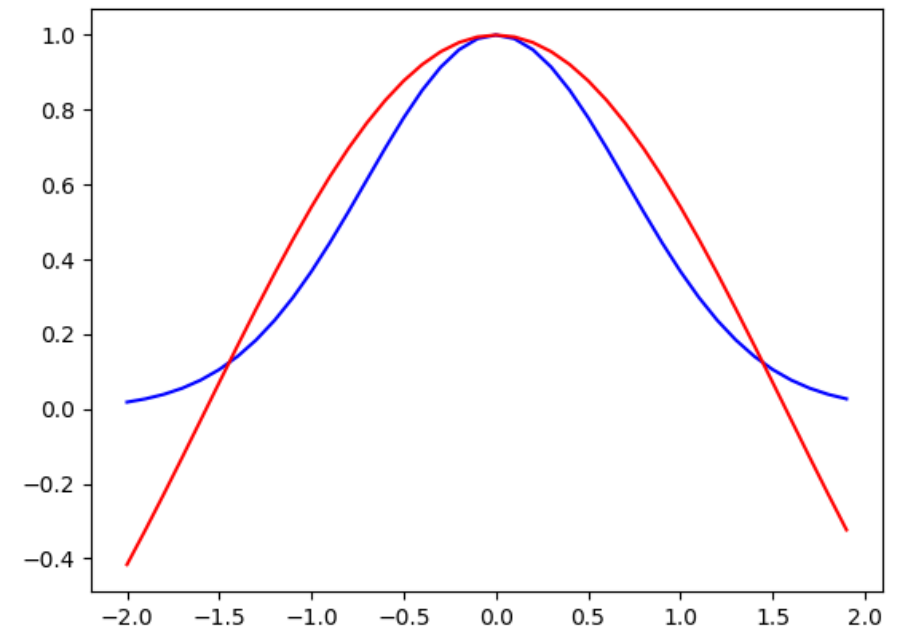


Gráficos de diversos vetores (cores diferentes)

Exemplo: Fazer o gráfico de $y(x) = e^{-x^2}$, $z(x) = \cos(x)$ $x > -2$

N = 40 termos

```
1 import numpy as np
2 import matplotlib.pyplot as fig
3 from math import cos, exp
4
5 n=int(input("número de termos = "))
6 x=np.zeros(n)
7 y=np.zeros(n)
8 z=np.zeros(n)
9 for i in range(n):
10     x[i]=-2+i*0.1
11     y[i]=exp(-x[i]**2)
12     z[i]=cos(x[i])
13 fig.plot(x,y,'-b',x,z,'-r')
```



Caracteres para formatação de gráficos

Caracteres indicadores de cores no *pyplot*

Cor	Caractere
Amarelo	'y'
Azul	'b'
Vermelho	'r'
Verde	'g'
Preto	'k'
Branco	'w'
Magenta	'm'
Ciano	'c'

Caracteres indicadores dos tipos de linha

Tipo de linha	Caractere
Linha cheia	' - '
Linha tracejada	' - - '
Linha traço-ponto	' - . '
Linha pontilhada	' : '

Caracteres indicadores dos tipos de marcadores

Tipo de marcador	Caractere
Ponto	' . '
Circulo	' o '
Triângulo	' v '
Triângulo	' ^ '
Triângulo	' < '
Triângulo	' > '
Quadrado	' s '
Pentágono	' p '
Estrela	' * '