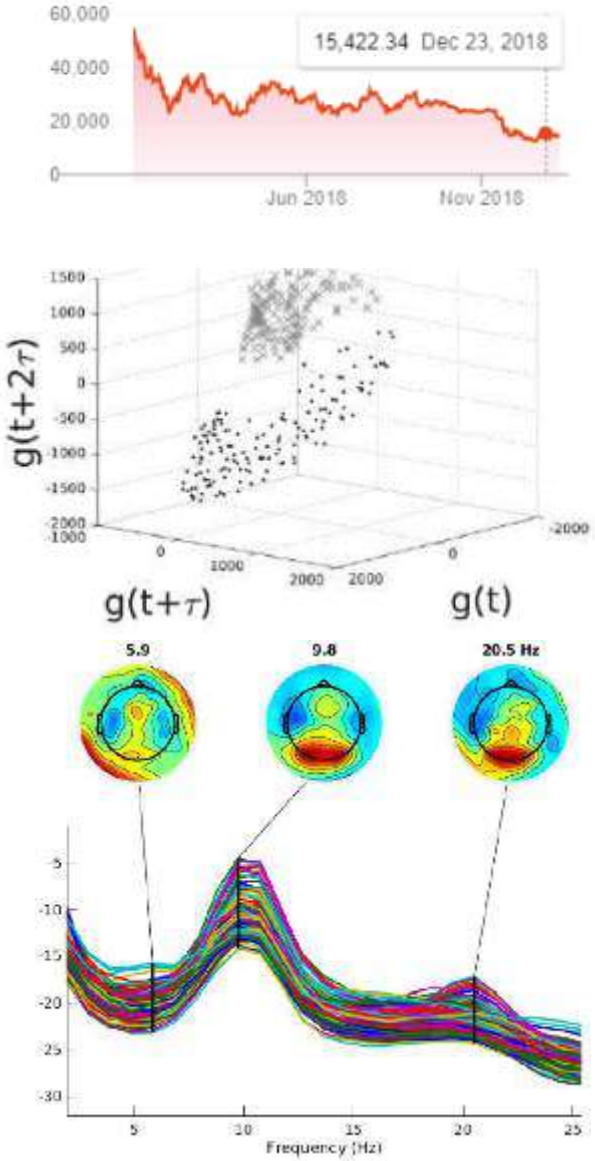


UNICAMP

Instituto de Computação

MC102: Algoritmos e Programação de Computadores (turmas 4,5,6,7)
Extra – Fazendo gráficos com Python

ARTHUR VALENCIO

Pós-doutorando IC/Unicamp

Fapesp CEPID NeuroMat

Campinas, 7 de Abril de 2020

>>Revisão de listas

```
minha_lista=[elemento1, elemento2, elemento3, elemento4]
```

>>Revisão de listas

Acessando elementos

```
minha_lista=[elemento1, elemento2, elemento3, elemento4]
```

Ordem direta

```
print(minha_lista[0])  <-- imprime elemento1
print(minha_lista[1])  <-- imprime elemento2
print(minha_lista[2])  <-- imprime elemento3
print(minha_lista[3])  <-- imprime elemento4
print(minha_lista[4])  <-- ERRO
```

Ordem inversa

```
print(minha_lista[-1])  <-- imprime elemento4
print(minha_lista[-2])  <-- imprime elemento3
print(minha_lista[-3])  <-- imprime elemento2
print(minha_lista[-4])  <-- imprime elemento1
```

>>Revisão de listas

Adicionando ou removendo elementos

```
minha_lista=[]
```
← Cria uma lista vazia

```
minha_lista.append(elemento1)
```
← Adiciona elemento1

```
minha_lista.append(elemento2)
```
← Adiciona elemento2

```
minha_lista.append(elemento3)
```
← Adiciona elemento3

```
print(minha_lista)
```
← Imprime a lista, isto é, mostra elemento1, elemento2 e elemento3 no terminal

```
minha_lista.pop(1)
```
← Remove item no índice [1], ou seja, elemento2

```
print minha_lista
```
← Agora, ao imprimir a lista, mostra só elemento1 e elemento3

>>Matplotlib

Agora que você já sabe listas, estamos prontos para usar um pacote poderoso de Python para construção de gráficos: **Matplotlib**

O Matplotlib foi construido com base no Matlab em 2003

>>Matplotlib: instalação

Para começar a usar as funcionalidades de construção de gráficos, digite a seguinte linha no cabeçalho do seu programa (ou no terminal python):

```
from matplotlib import pyplot
```

ou

```
import matplotlib.pyplot
```

Se aparecer uma mensagem de erro, é preciso instalar o pacote. Nesse caso vá ao Prompt de Comando do Windows ou Terminal do Linux/Mac e digite:

```
pip3 install matplotlib
```

>>Gráficos de linha

```
from matplotlib import pyplot
```

```
y=[1, 3, 2.5, -5, 3.14, -2.7, 0.01]
```

Tenha em suas mãos a sua lista de dados

```
pyplot.plot(y)
```

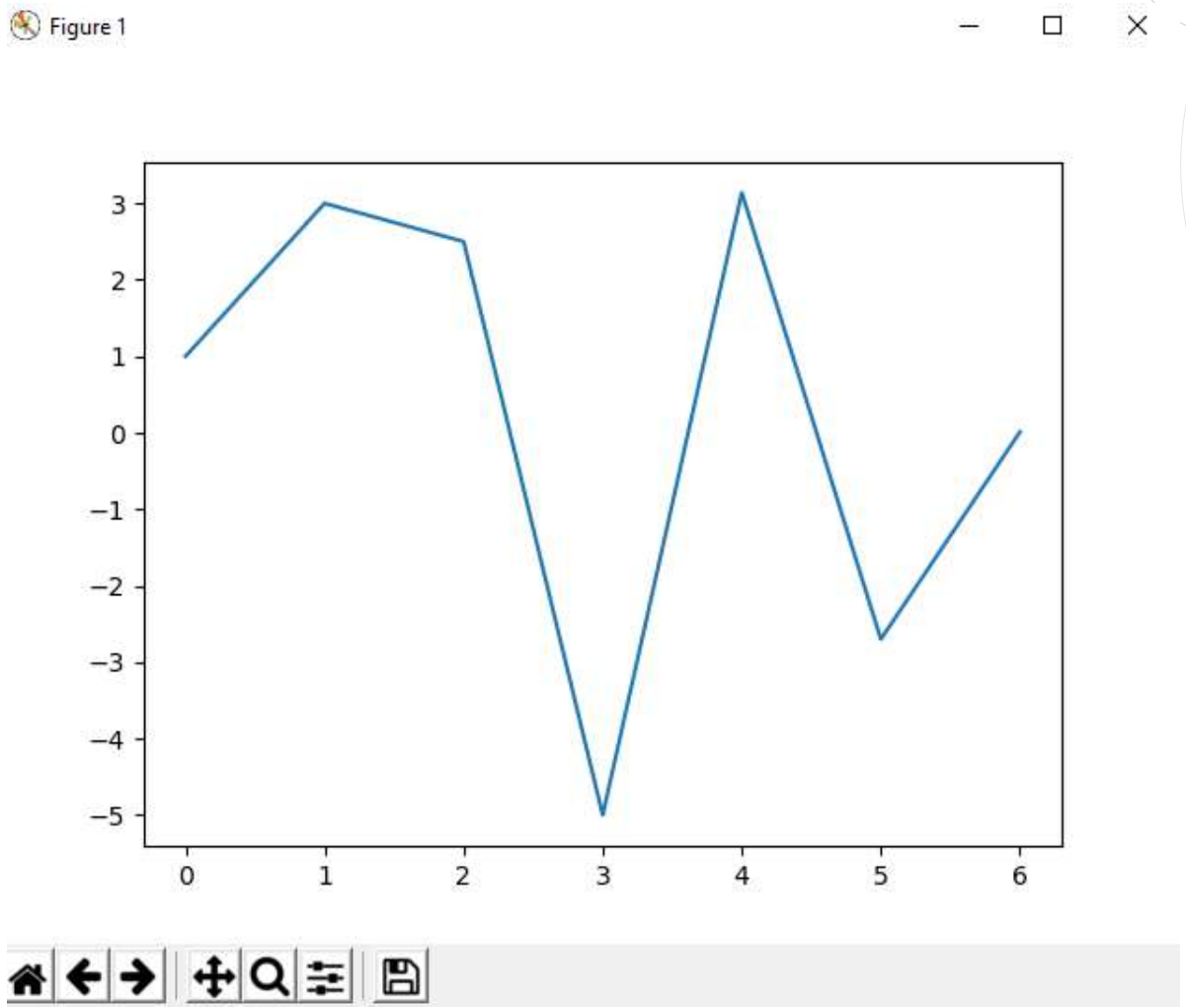
Constrói o gráfico

```
pyplot.show()
```

Cria janela mostrando o gráfico para o usuário

>>Gráficos de linha

Este é o seu resultado:



>>Gráficos de linha

Agora vamos adicionar mais detalhes

```
from matplotlib import pyplot

x=[1, 3, 3.5, 5, 7.14, 12.7, 20.01]
y=[1, 3, -2.5, -5, 3.14, -2.7, 0.01]

pyplot.plot(x, y, 'ro:')
pyplot.xlabel("Eixo x")
pyplot.ylabel("Eixo y")
pyplot.title("Meu Gráfico")
pyplot.show()
```

Faremos agora um gráfico os dados do eixo x não são igualmente espaçados

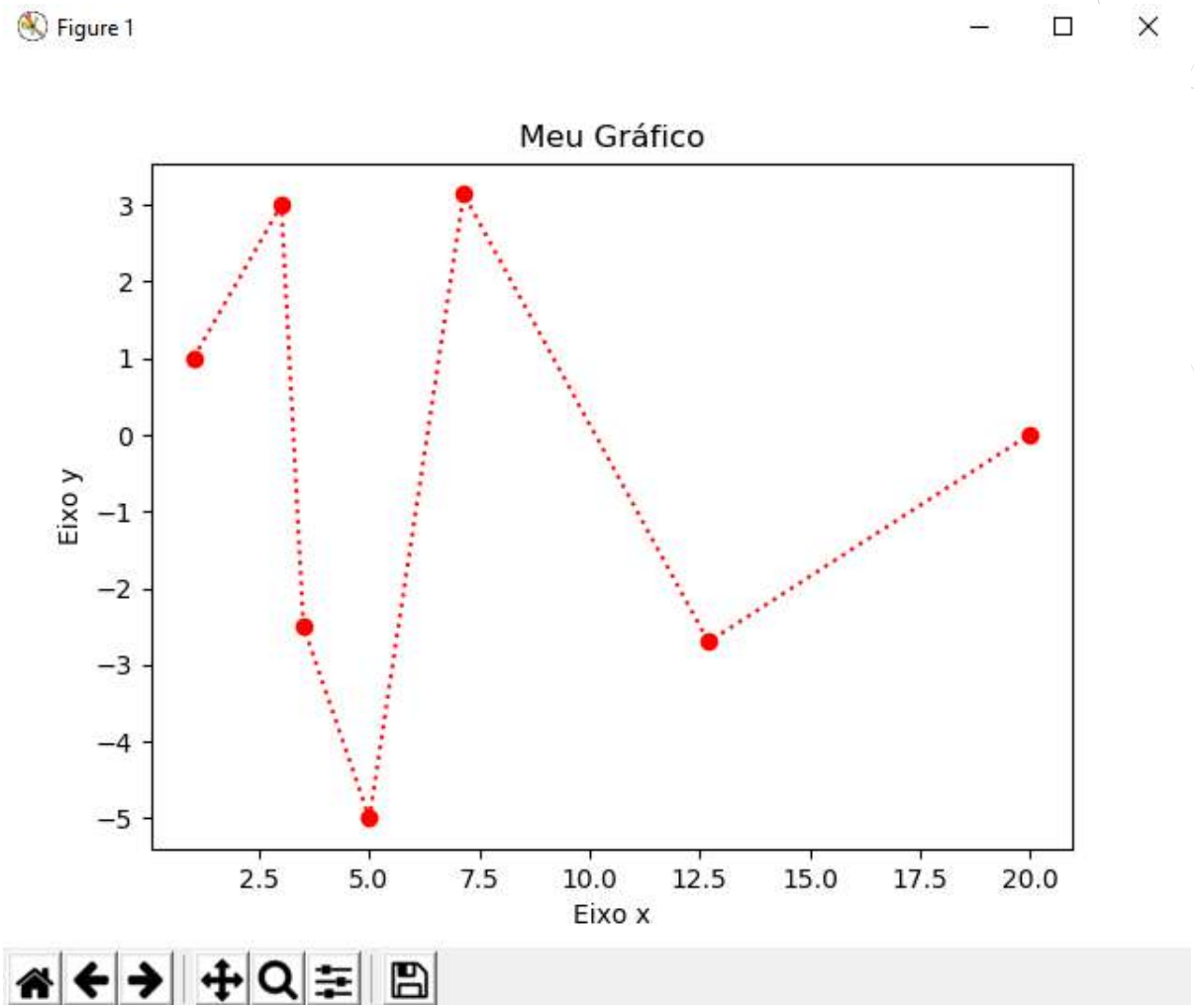
Escolhemos que o gráfico será colorido de vermelho (r), que os pontos dos dados serão preenchidos com círculos (o) e que a linha conectando os pontos será pontilhada (:)

Adicionamos rótulos aos eixos

Adicionamos título ao gráfico

>>Gráficos de linha

Este é o seu resultado:



>>Opções mais utilizadas

Cor:

- **b** azul (blue)
- **g** verde (green)
- **r** vermelho (red)
- **c** ciano (cyan)
- **m** magenta
- **y** amarelo (yellow)
- **k** preto (black)
- **w** branco (white)

Tipo de linha:

- **-** contínua
- **--** tracejada
- **:** pontilhada
- **-.** traço e ponto

Tipo de marcador:

- **.** ponto
- **,** único pixel
- **o** círculo
- **s** quadrado (square)
- **d** losango/diamante
- **x** x
- **+** mais
- **^** triângulo
- **1** tripé
- ***** estrela
- **h** hexágono

>>Vários gráficos em um

```
from matplotlib import pyplot

x1=[1, 3, 4, 5, 7, 8, 10]
y1=[2, 6, 8, 10, 14, 16, 20]
x2=[1, 2, 5, 6, 8, 9, 10]
y2=[1, 4, 25, 36, 64, 81, 100]
x3=[1, 3, 5, 7, 9, 10, 11]
y3=[0.1,2.7, 12.5, 34.3, 72.9, 100, 133]

pyplot.plot(x1, y1, 'ro:',x2, y2, 'bs--',x3, y3, 'g^-.')
pyplot.xlabel("Eixo x")
pyplot.ylabel("Eixo y")
pyplot.title("Meu Gráfico")
pyplot.legend(["y=2x","y=x^2", "y=x^3/10"])
pyplot.show()
```

Informando as novas variáveis ao fim do próprio plot

Criando barra de legenda

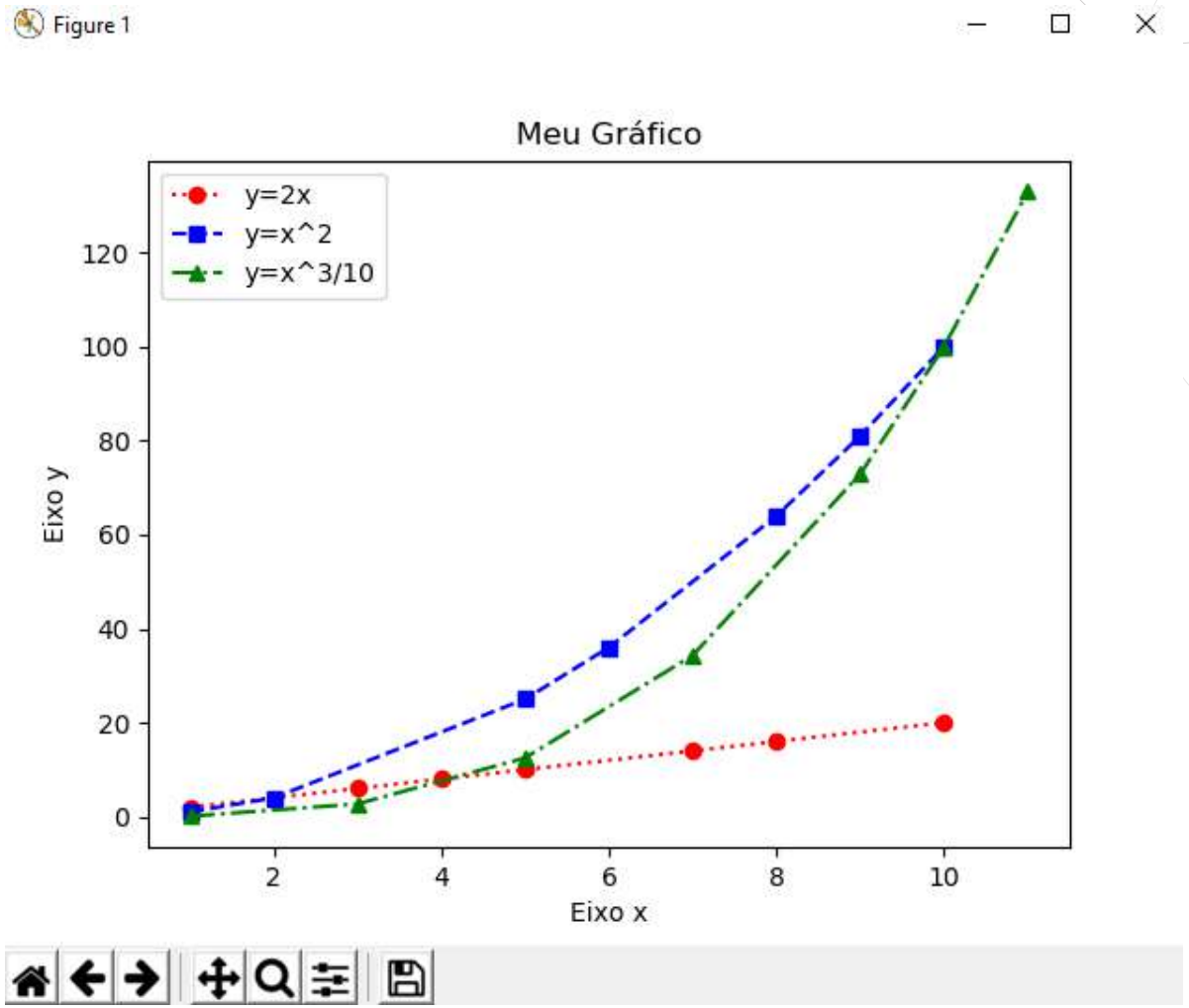
>>Vários gráficos em um

```
from matplotlib import pyplot

x1=[1, 3, 4, 5, 7, 8, 10]
y1=[2, 6, 8, 10, 14, 16, 20]
x2=[1, 2, 5, 6, 8, 9, 10]
y2=[1, 4, 25, 36, 64, 81, 100]
x3=[1, 3, 5, 7, 9, 10, 11]
y3=[0.1,2.7, 12.5, 34.3, 72.9, 100, 133]

pyplot.plot(x1, y1, 'ro:' )
pyplot.plot(x2, y2, 'bs--' )
pyplot.plot(x3, y3, 'g^-.')
pyplot.xlabel("Eixo x")
pyplot.ylabel("Eixo y")
pyplot.title("Meu Gráfico")
pyplot.legend(["y=2x","y=x^2", "y=x^3/10"])
pyplot.show()
```

Alternativamente, você pode realizar vários **plot()**, mas finalizando com um único **show()**

>>Gráficos de linha

Este é o seu resultado:



>>Gráficos de pizza

Caso vocês queiram separar um (ou mais) dos valores da pizza, para dar ênfase (opcional)

```
from matplotlib import pyplot

x=[1, 3, 4, 2.5, 3.5]
nomes=["Valor 1", "Valor 2", "Valor 3", "Valor 4", "Valor 5"]
separar=[0, 0, 0.1, 0, 0]

pyplot.pie(x, labels=nomes, explode=separar, autopct='%1.1f%%', startangle=90)
pyplot.axis('equal')
pyplot.show()
```

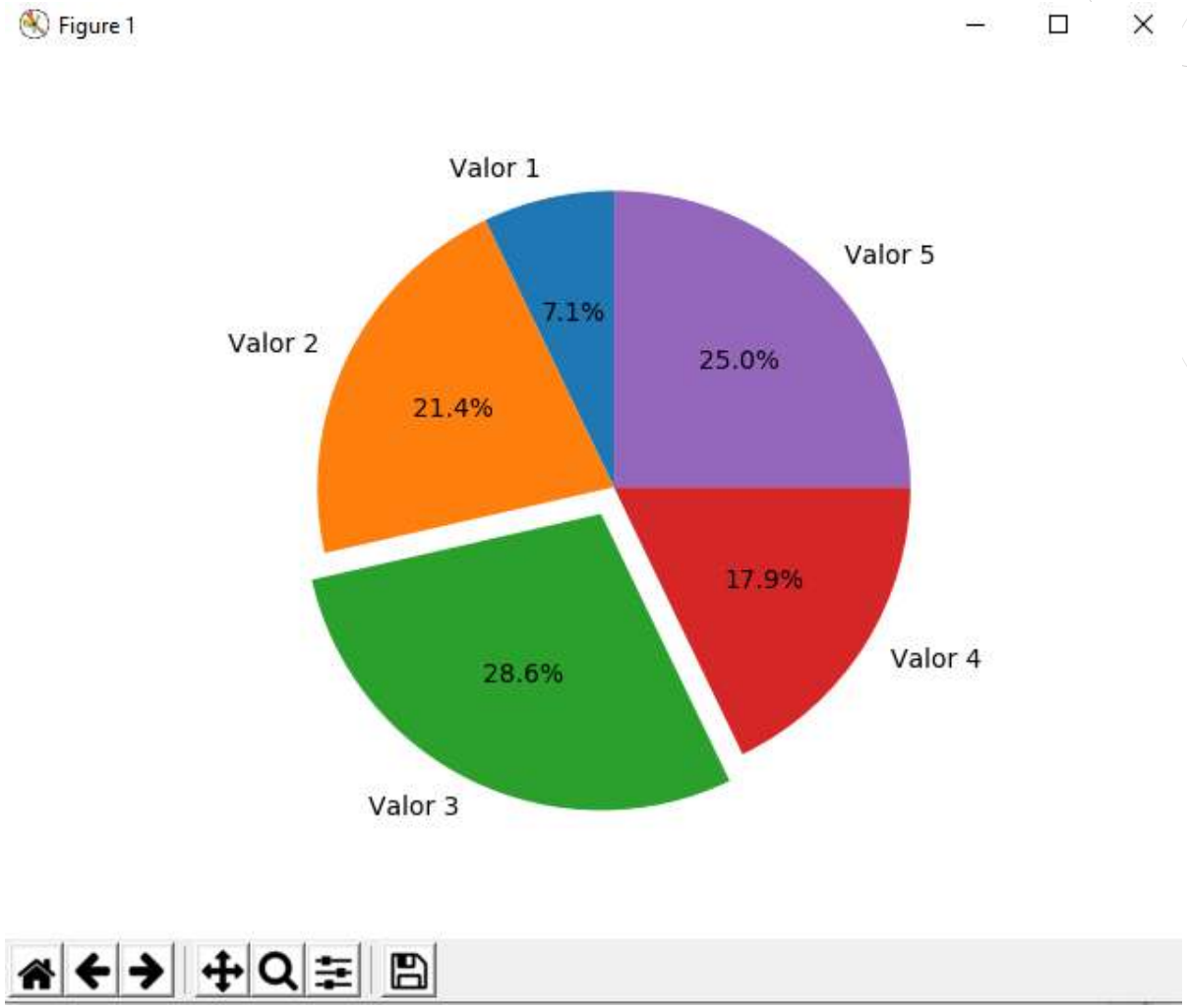
Garante que gráfico será desenhado como círculo (opcional, mas altamente recomendado)

Coloca porcentagem na figura (no caso, um float com 1 casa decimal)

Para que o ângulo de visão seja vertical, não inclinado (opcional)

>>Gráficos de pizza

Este é o seu resultado:



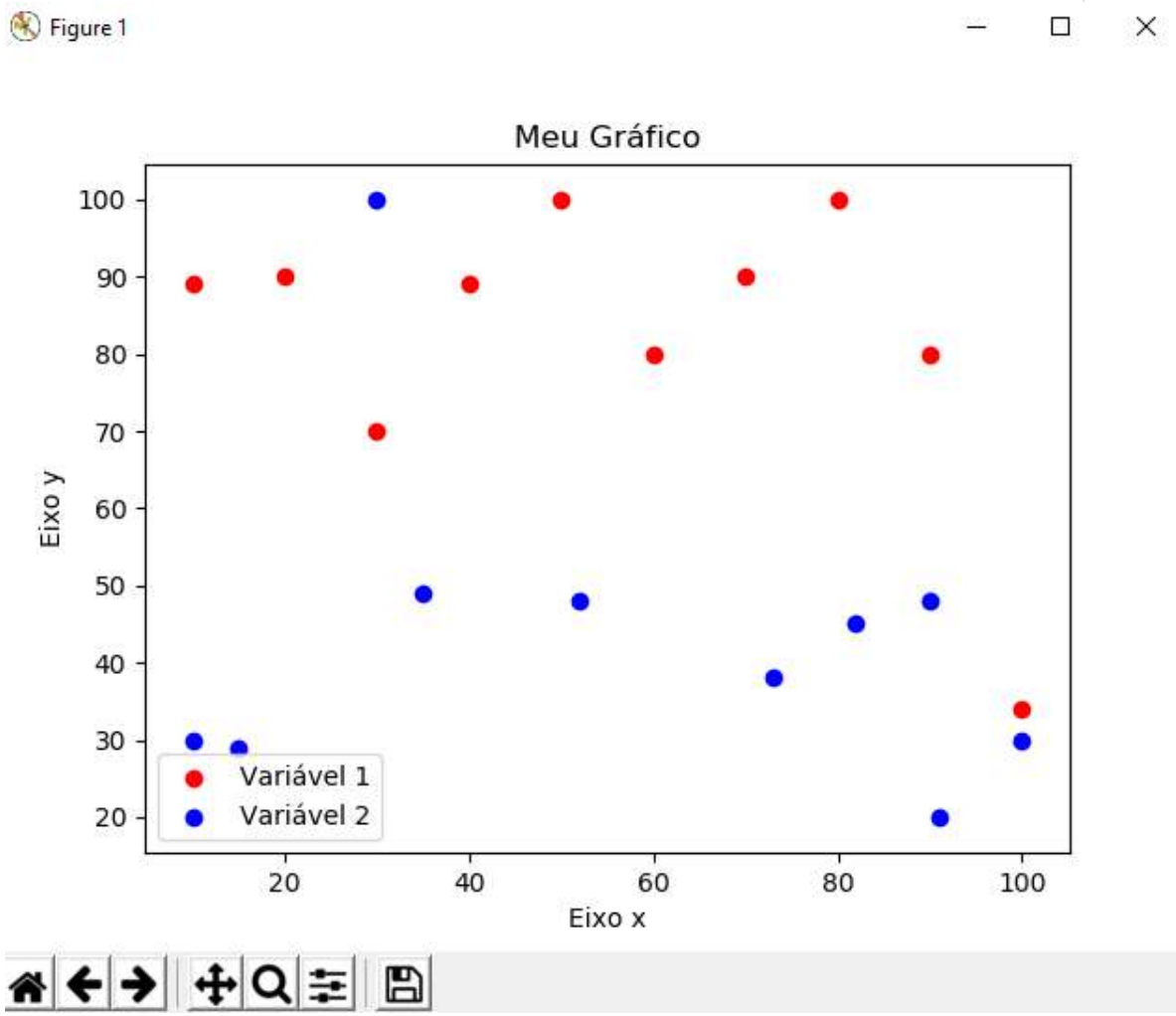
>>Gráficos de dispersão

```
from matplotlib import pyplot

x1 = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
y1 = [89, 90, 70, 89, 100, 80, 90, 100, 80, 34]
x2= [10, 15, 35, 52, 30, 90, 73, 82, 91, 100]
y2 = [30, 29, 49, 48, 100, 48, 38, 45, 20, 30]

pyplot.scatter(x1, y1, color='r')
pyplot.scatter(x2, y2, color='b')
pyplot.xlabel('Eixo x')
pyplot.ylabel('Eixo y')
pyplot.title('Meu Gráfico')
pyplot.legend(['Variável 1', 'Variável 2'])
pyplot.show()
```

Defino a cor da primeira variável como vermelha e da segunda como azul

>>Gráficos de dispersão

Este é o seu resultado:



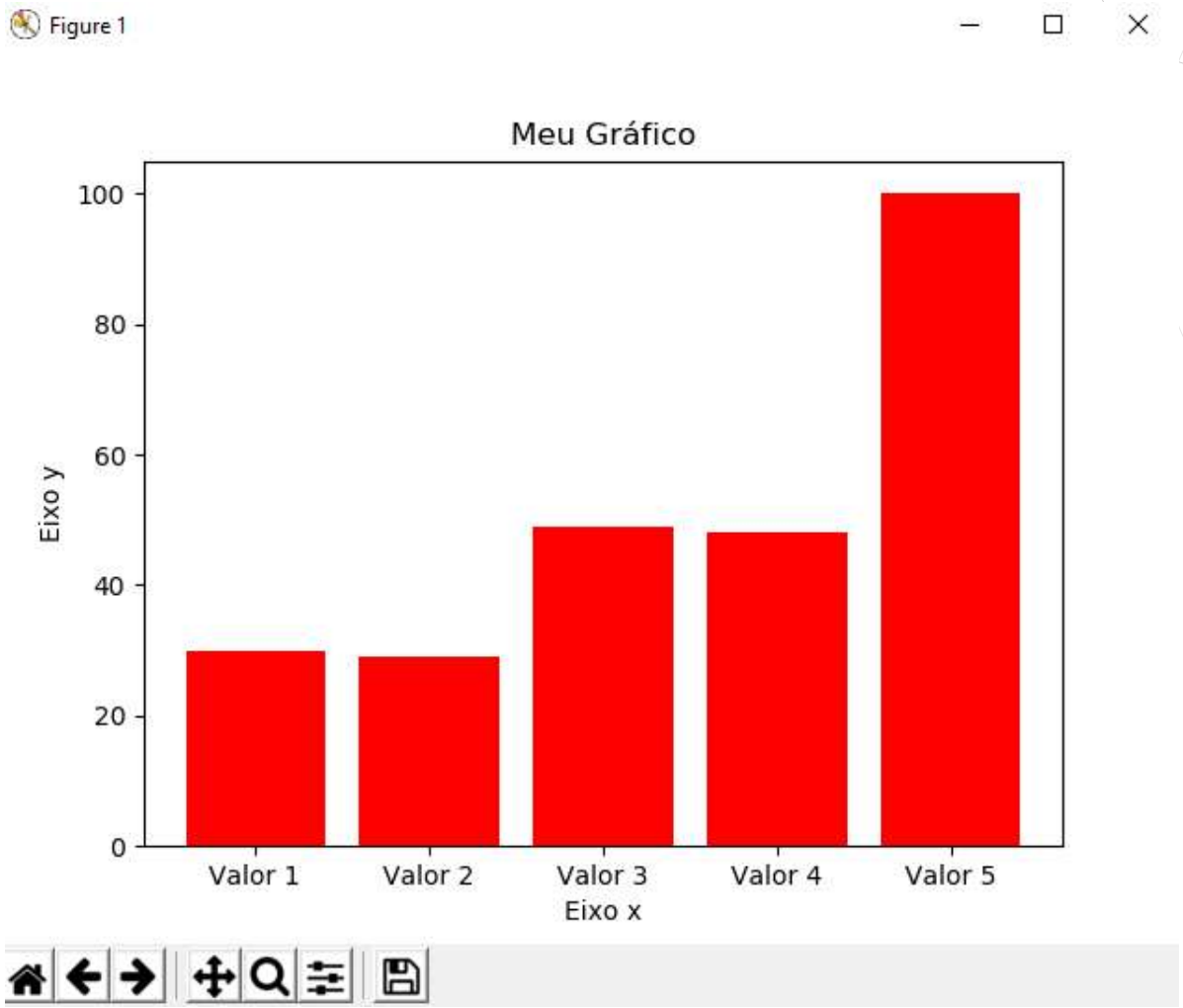
>>Gráficos de barra

```
from matplotlib import pyplot

x = [30, 29, 49, 48, 100]
nomes=["Valor 1", "Valor 2", "Valor 3", "Valor 4", "Valor 5"]

pyplot.bar(nomes, x, color='r')
pyplot.xlabel('Eixo x')
pyplot.ylabel('Eixo y')
pyplot.title('Meu Gráfico')
pyplot.show()
```

Note que você coloca o rótulo dos eixos primeiro e os valores depois

>>Gráficos de barra

Este é o seu resultado:



>>Histograma

```
import random
from matplotlib import pyplot

x = []
i=0
while i<100000:
        x.append(random.normalvariate(10,5) )
        i=i+1

pyplot.hist(x, bins=50, facecolor='g', density='True')
pyplot.xlabel('Valores')
pyplot.ylabel('Probabilidade')
pyplot.title('Meu Gráfico')
pyplot.show()
```

Distribuição normal de média 10 e desvio 5

Criando uma variável (distribuição normal com 100000 pontos)

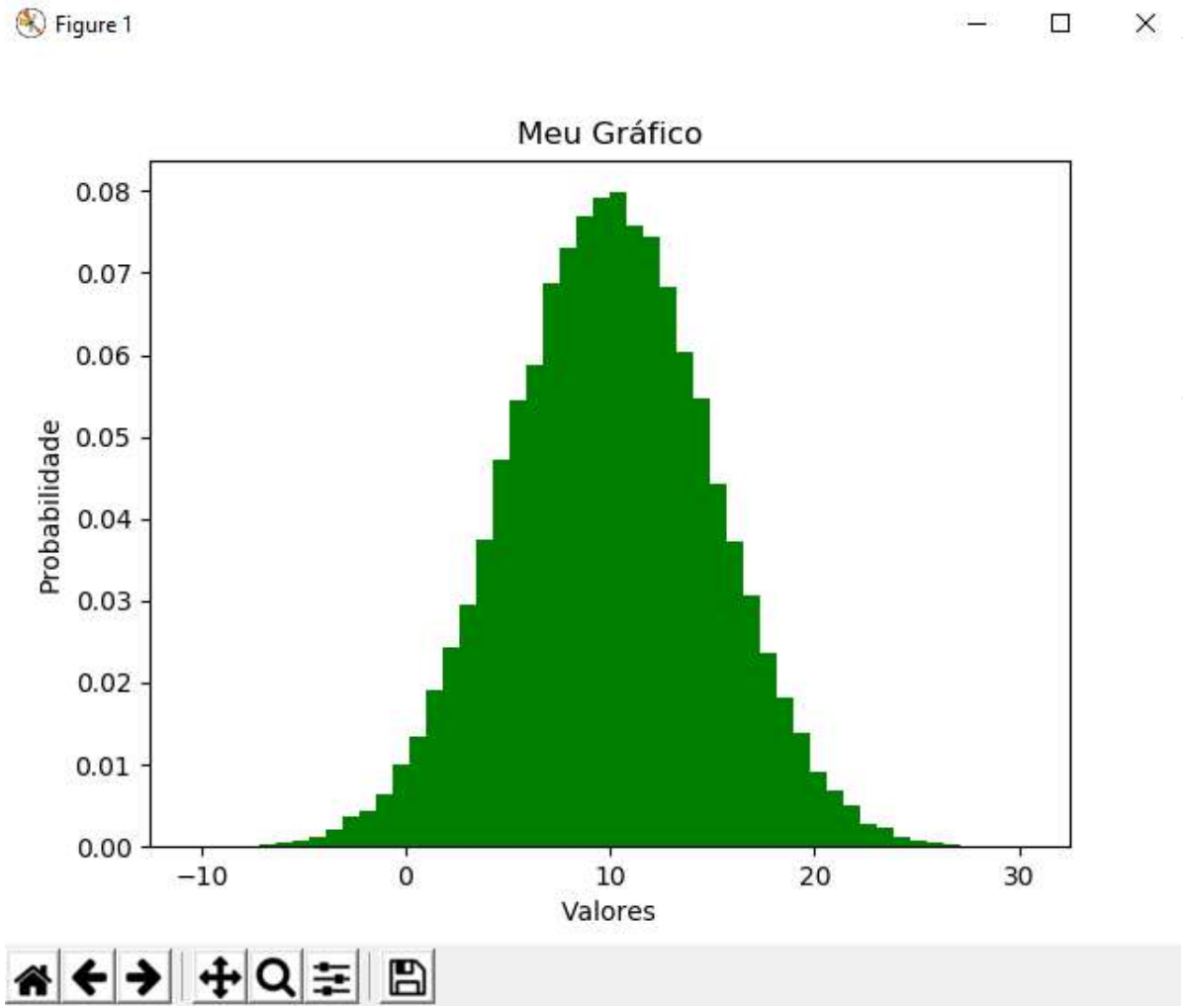
Defino 50 intervalos de valores para a análise (bins)

A cor do gráfico será verde

Quero o eixo y como densidade de probabilidade (do contrarário ele só informa o número de pontos em cada intervalo)

>>Histograma

Este é o seu resultado:

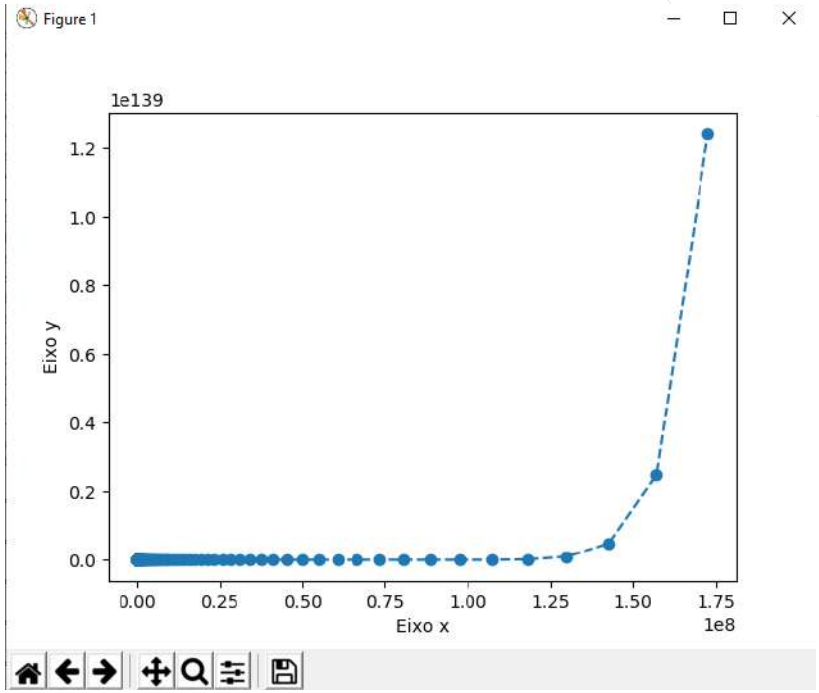


>>Escala logarítimica

Considere este código para uma curva de crescimento rápido

Aqui, a cada passo o valor no eixo fica 10% maior do que o anterior.
Já o eixo y é multiplicado por 5 a cada passo.

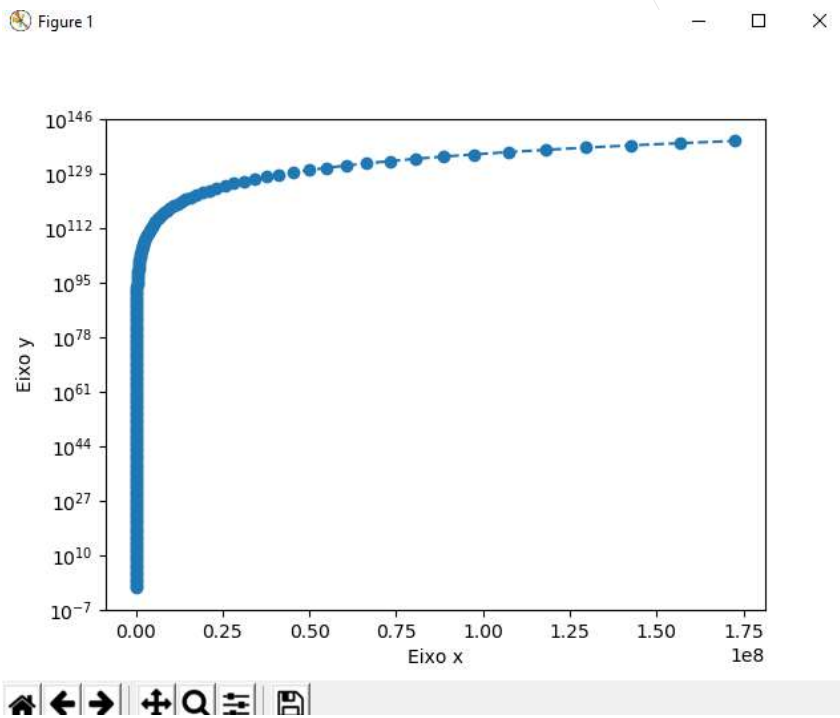
```
import random
from matplotlib import pyplot

x = [1]
y = [1]
i=1
while i<200:
    x.append(x[i-1]*1.1)
    y.append(y[i-1]*5)
    i=i+1

pyplot.plot(x,y)
pyplot.xlabel('Eixo x')
pyplot.ylabel('Eixo y')
pyplot.show()
```



>>Escala logarítimica

```
import random
from matplotlib import pyplot

x = [1]
y = [1]
i=1
while i<200:
    x.append(x[i-1]*1.1)
    y.append(y[i-1]*5)
    i=i+1

pyplot.plot(x,y)
pyplot.xlabel('Eixo x')
pyplot.ylabel('Eixo y')
pyplot.yscale('log')
pyplot.show()
```

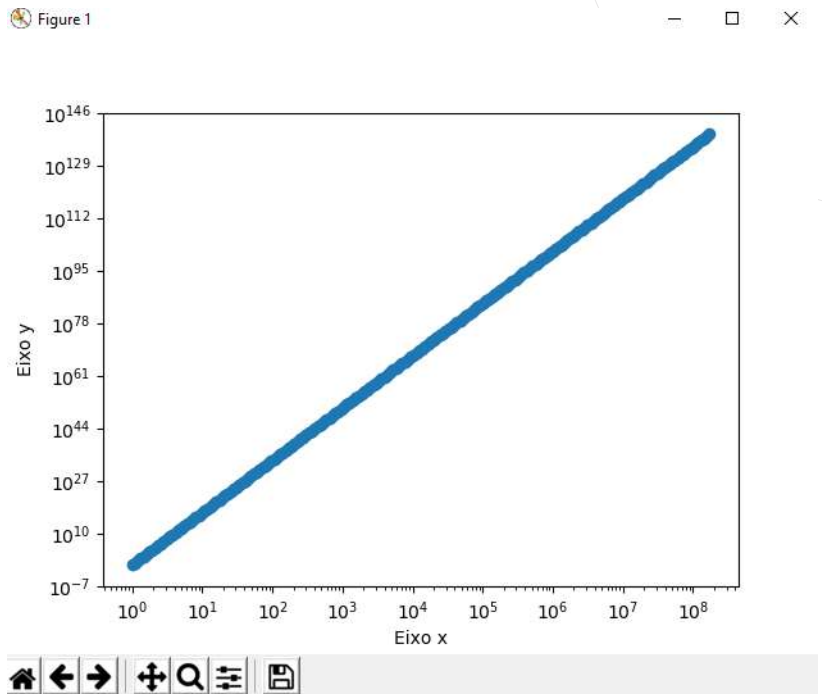
Adicionamos essa linha para ajustarmos o eixo y para a escala logarítmica (gráfico semilog)



>>Escala logarítimica

```
import random
from matplotlib import pyplot

x = [1]
y = [1]
i=1
while i<200:
    x.append(x[i-1]*1.1)
    y.append(y[i-1]*5)
    i=i+1

pyplot.plot(x,y)
pyplot.xlabel('Eixo x')
pyplot.ylabel('Eixo y')
pyplot.xscale('log')
pyplot.yscale('log')
pyplot.show()
```

Fazendo o mesmo para o eixo x, temos o gráfico log-log



>>Multiplas figuras

Vamos fazer com que as três figuras do exemplo anterior sejam geradas pelo mesmo código:

```
import random
from matplotlib import pyplot

x = [1]
y = [1]
i=1
while i<200:
    x.append(x[i-1]*1.1)
    y.append(y[i-1]*5)
    i=i+1

pyplot.figure(1)
pyplot.plot(x,y)

pyplot.figure(2)
pyplot.plot(x,y)
pyplot.yscale('log')

pyplot.figure(3)
pyplot.plot(x,y)
pyplot.xscale('log')
pyplot.yscale('log')
pyplot.show()
```

Os comandos **figure** geram diferentes telas de figuras para cada um dos plots que vamos fazer

Não se esqueça de terminar mostrando as figuras geradas para o usuário

>>Multiplos gráficos em única figura (subplot)

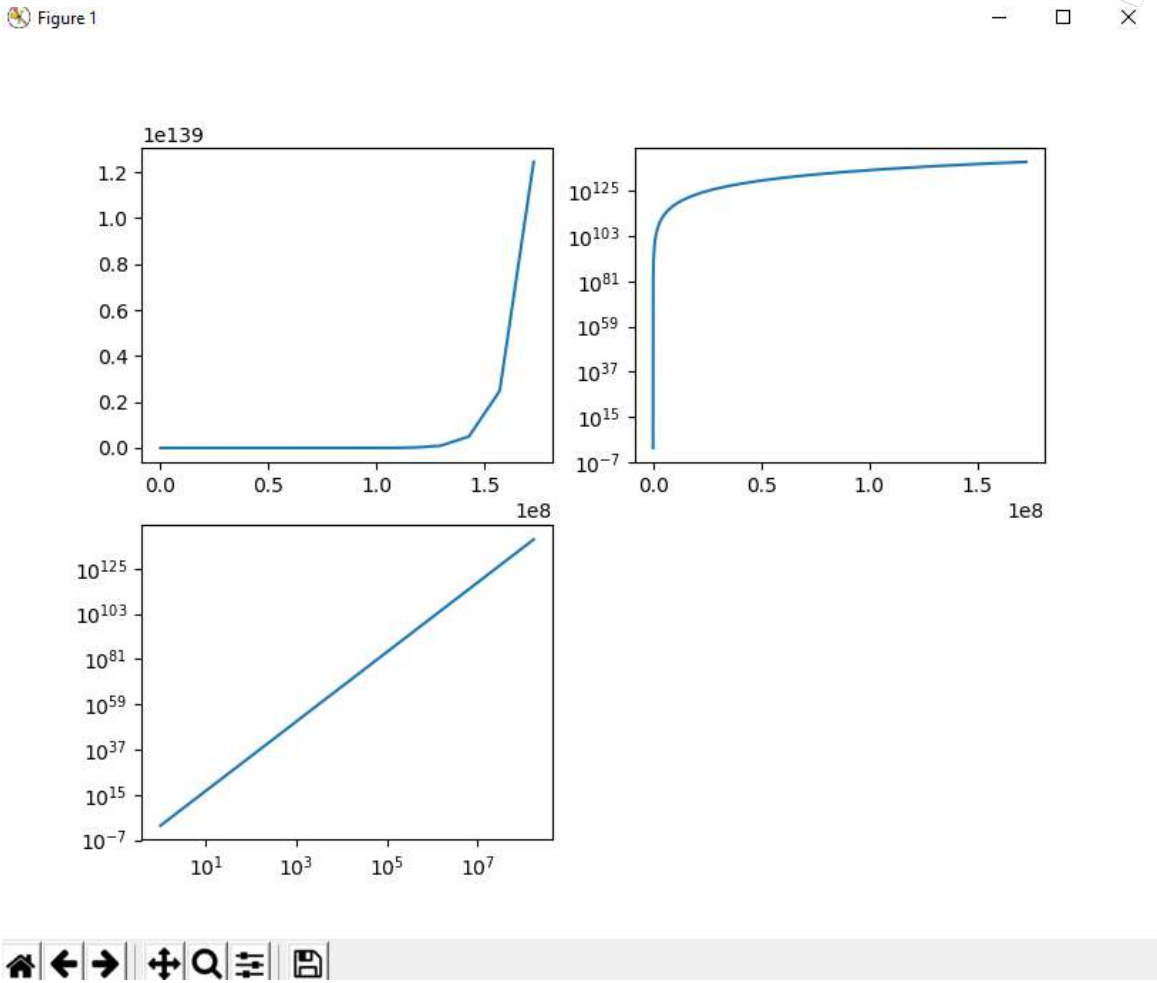
```
import random
from matplotlib import pyplot

x = [1]
y = [1]
i=1
while i<200:
    x.append(x[i-1]*1.1)
    y.append(y[i-1]*5)
    i=i+1

pyplot.subplot(2,2,1)
pyplot.plot(x,y)

pyplot.subplot(2,2,2)
pyplot.plot(x,y)
pyplot.yscale('log')

pyplot.subplot(2,2,3)
pyplot.plot(x,y)
pyplot.xscale('log')
pyplot.yscale('log')
pyplot.show()
```

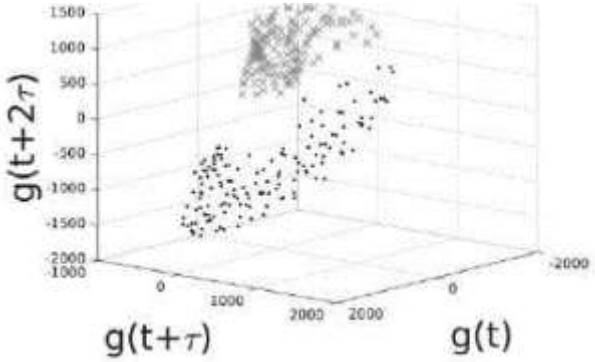
subplot (no. de linhas, no. colunas, índice):

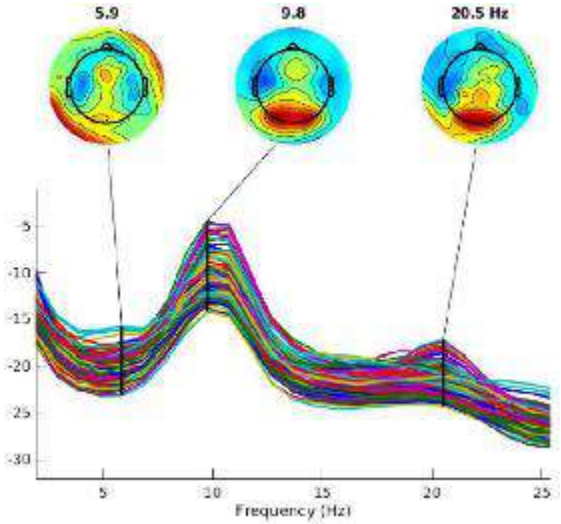
Significa que, ao chamar **subplot(2,2,1)** , estou falando que na figura existe um grid de gráficos com 2 linhas e 2 colunas, e que o gráfico que estou plotando logo em seguida será o 1º.

>>Multiplos gráficos em única figura (subplot)

Este é o seu resultado:







UNICAMP

Instituto de Computação

That´s all folks!

arthur_valencio@physics.org

http://www.arthurvalencio.com/mc102

http://www.ic.unicamp.br/~mc102