

# Curso de Iniciação à Programação em Python para Profissionais de Saúde



Luís Vieira e José Ferrão

Instituto Nacional de Saúde Doutor Ricardo Jorge

2-11 Dezembro 2024

# Sessão 5: Módulos

- **Conteúdos:**
  - Módulos
  - Pacote matplotlib
    - Módulo pyplot
  - Exercícios

# Módulos

- O *Python* tem a funcionalidade de colocar definições de funções num ficheiro para serem usadas em *scripts* ou em instâncias interactivas do interpretador. A este ficheiro dá-se o nome de **módulo**
- Um módulo contém definições e declarações do *Python*. O nome do ficheiro é o nome do módulo a que está acrescentado o sufixo *.py*.
- Os módulos são importados usando a declaração *import*, que normalmente se escreve no topo do *script* ou directamente na *Shell* do *Python*. Por exemplo, para importar o módulo de funções matemáticas\* na *Shell* usa-se:  
  

```
>>> import math
```
- Em alternativa, podem importar-se directamente funções contidas num dado módulo. Por exemplo, pode importar-se a função *ceil()* do módulo **math**:

```
>>> from math import ceil
```

# Módulos

- É também possível importar todo o conteúdo de um dado módulo usando:

```
>>> from math import *
```

- Os nome dos módulos importados podem ser alterados pelo utilizador se for conveniente. Por exemplo, o módulo **math** pode ser importado com a designação '*m*':

```
>>> import math as m
```

- A informação sobre as funções importadas pode ser obtida usando a função *help*(). Por exemplo:

```
>>> help(ceil)
```

- A função *help*() pode também ser usada para se conhecer todo o conteúdo do módulo:

```
>>> help(math)
```

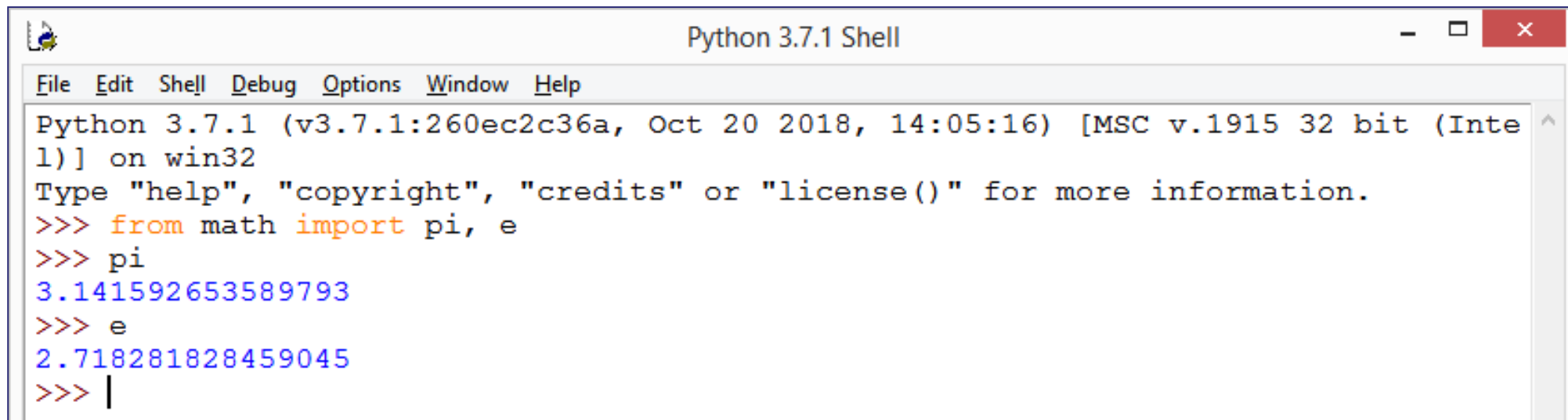
# Módulos

- Além das funções, o módulo **math** também contém algumas constantes matemáticas que também podem ser importadas, como por exemplo os valores de 'pi' e 'e':

```
>>> from math import pi, e
```

```
>>> pi
```

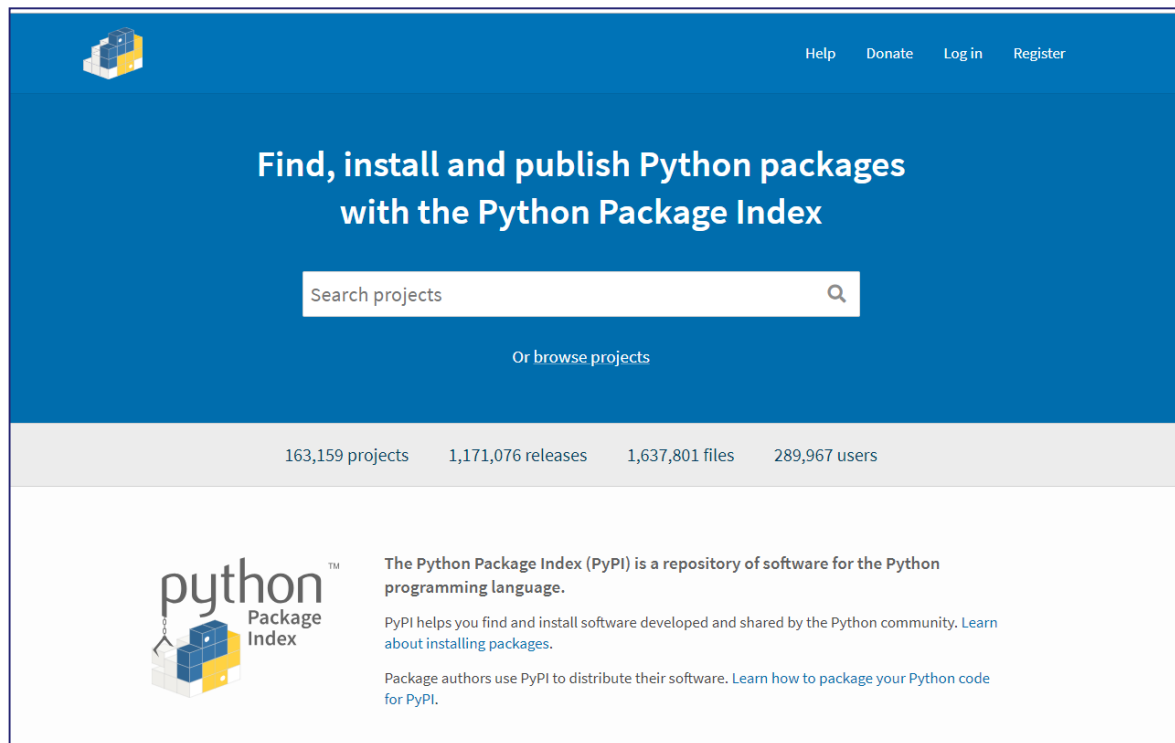
```
>>> e
```



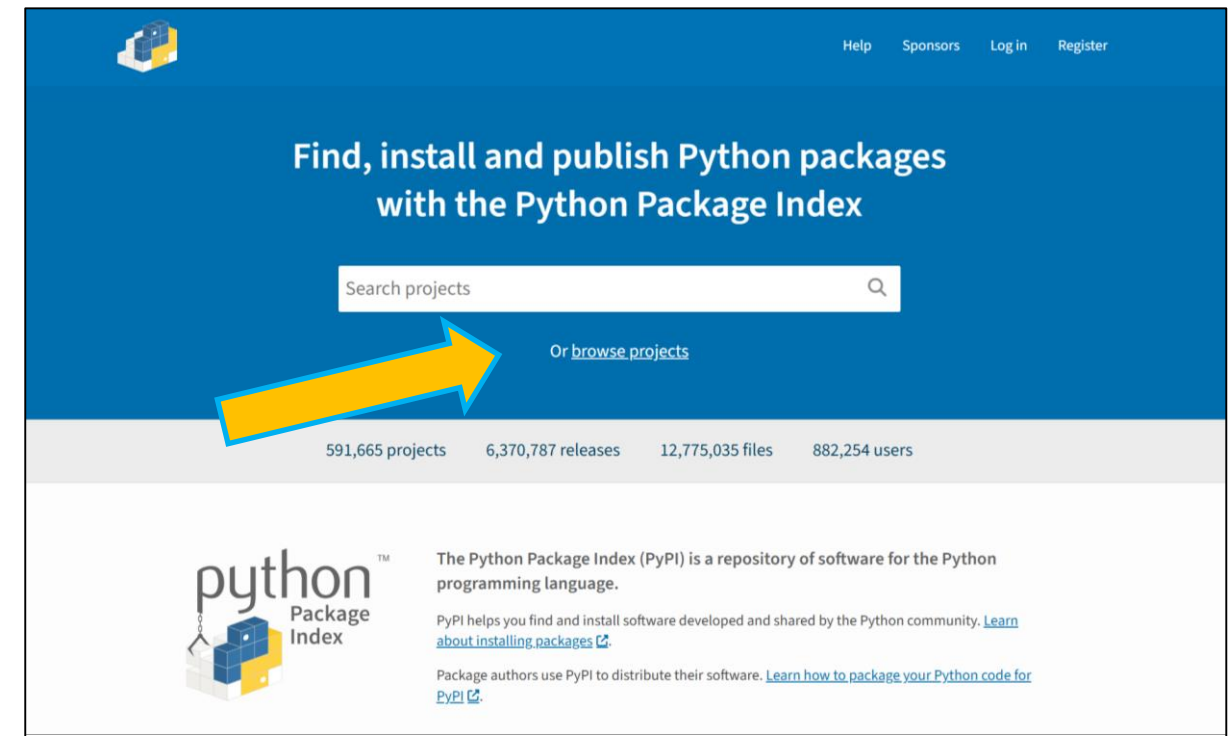
```
Python 3.7.1 Shell
File Edit Shell Debug Options Window Help
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:05:16) [MSC v.1915 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> from math import pi, e
>>> pi
3.141592653589793
>>> e
2.718281828459045
>>> |
```

# Python Package Index

- O *Python Package Index* é um repositório de pacotes de *software open-source* licenciado, disponível para todos os utilizadores do *Python*




<https://pypi.org/>, acedido em 30/12/2018



<https://pypi.org/>, acedido em 09/12/2024

# Python Package Index



[Help](#)
[Sponsors](#)
[Log in](#)
[Register](#)

Filter by [classifier](#)


☒ Framework
   
☒ Topic

- ☐ Adaptive Technologies
- ☐ Artistic Software
- ☐ Communications
  - ☐ BBS
  - ☐ Chat
    - ☐ ICQ
    - ☐ Internet Relay Chat
    - ☐ Unix Talk
  - ☐ Conferencing
  - ☐ Email
    - ☐ Address Book
    - ☐ Email Clients (MUA)
    - ☐ Filters
    - ☐ Mail Transport Agents
    - ☐ Mailing List Servers
    - ☐ Post-Office
      - ☐ IMAP
      - ☐ POP3
  - ☐ FIDO
  - ☐ Fax
  - ☐ File Sharing
    - ☐ Gnutella

2,799 projects with the selected classifiers


Order by Relevance

PROGRAMMING LANGUAGE :: PYTHON :: 3
TOPIC :: SCIENTIFIC/ENGINEERING :: BIO-INFORMATICS




**aiondata** Nov 20, 2024

A common data access layer for AI-driven drug discovery.




**alphaviz** Aug 2, 2022

A interactive Dashboard to explore mass spectrometry data.




**aop2db** Sep 8, 2022

AOP2DB - Python parser for converting importing adverse outcome pathway data into a relational database.




**AxionVueOpenAPI** Mar 30, 2023

This is a python wrapper for the Axion Vue windows app to use it headless



**AxionVueOpenAPI-different** Sep 20, 2024

This is a python wrapper for the Axion Vue windows app to use it headless



**bedqr** Aug 2, 2022



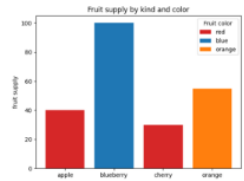
# matplotlib

- Neste curso vamos usar o pacote **matplotlib** do *Python*, que contém uma biblioteca de gráficos 2D (linhas, barras, dispersão, etc.) com qualidade para publicação de figuras em diferentes formatos
- O **matplotlib** contém um conjunto de módulos, cada um com várias funções associadas, que podem ser usados em *scripts* do *Python*
- Este pacote não vem pré-instalado com o *Python 3*, é necessário instalá-lo usando o programa *pip*.  
Abrir a linha de comandos do Windows (*cmd.exe*) e escrever:  
  
> *pip install matplotlib --user (\*)*
- Este comando instala o **matplotlib** e todas as dependências necessárias para usar os módulos do **matplotlib**
- Para usar o **matplotlib** após instalação, é necessário fechar e iniciar novamente o interpretador do *Python*

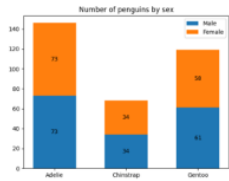
\* Nota: A opção - *--user* só é necessária se não tiver permissões de administrador



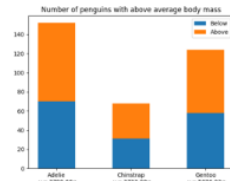
# matplotlib



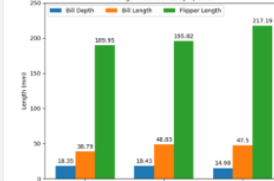
Bar color demo



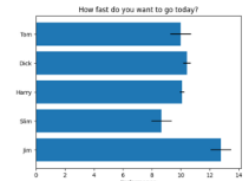
Bar Label Demo



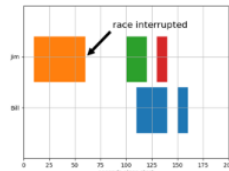
Stacked bar chart



Grouped bar chart with labels



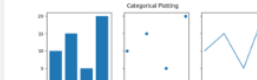
Horizontal bar chart



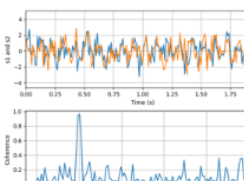
Broken Barh



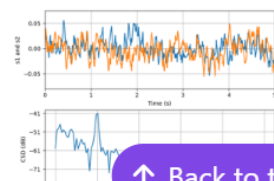
CapStyle



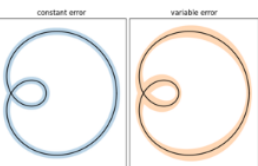
Plotting categorical variables



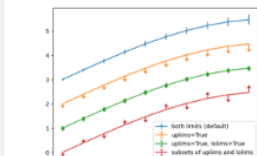
Plotting the



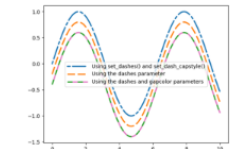
Cross spectral



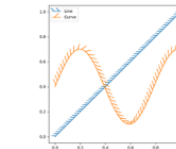
Curve with error



Errorbar limit



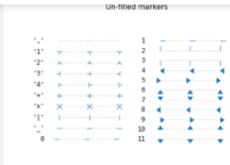
Customizing dashed line styles



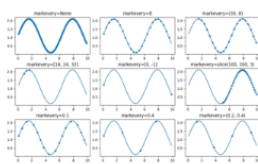
Lines with a ticked path effect



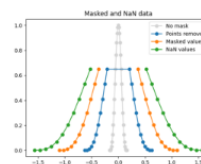
Linestyles



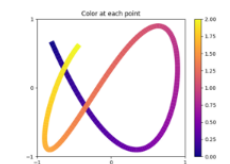
Marker reference



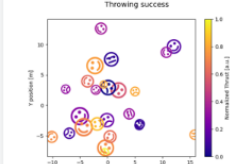
Markevery Demo



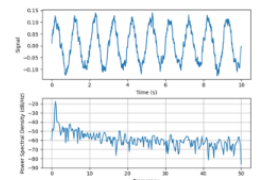
Plotting masked and NaN values



Multicolored lines



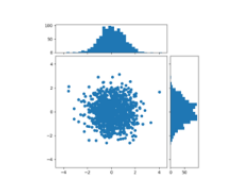
Mapping marker properties to multivariate data



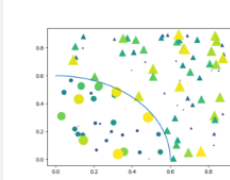
Power spectral density (PSD)



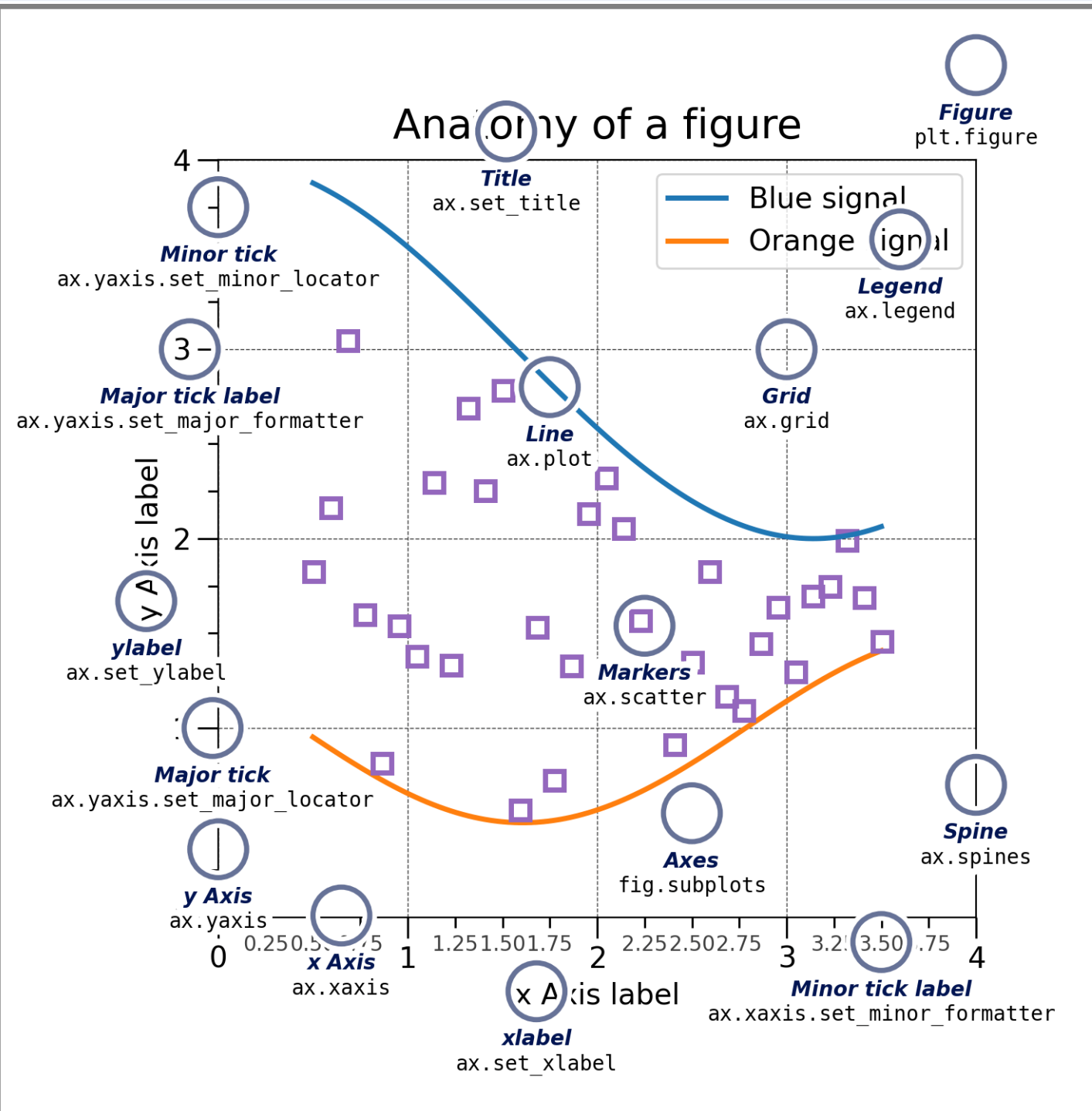
Scatter Demo2



Scatter plot with histograms



Scatter Masked



# matplotlib

## Componentes de uma figura

# Módulo pyplot

- Vamos usar o módulo **pyplot** com dados do ficheiro de doenças de declaração obrigatória
- O objectivo é criar uma figura que represente graficamente o número de doenças (em forma de linha) e de indivíduos (em forma de barras verticais), notificados em cada mês ao longo de um ano, usando o módulo **pyplot** do **matplotlib**
- O **módulo pyplot** contém um conjunto vasto de funções para criar uma figura, inserir um gráfico, carregar dados no gráfico, decorar a figura, etc.
- Para começar, vamos abrir um novo ficheiro no editor do *Python* e escrever os seguintes comandos para importar o módulo **pyplot**:

*from* matplotlib *import* pyplot *as* plt

# Módulo pyplot

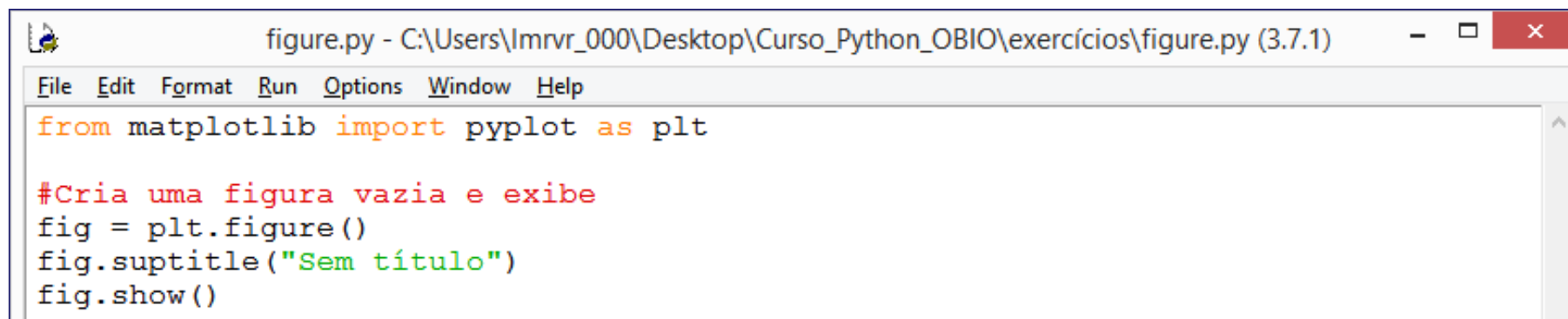
- De seguida, vamos criar uma figura vazia e dar-lhe um título provisório:

```
fig = plt.figure()
fig.suptitle ("Sem título")
```

- Para visualizar a figura numa janela à parte, escrevemos o seguinte comando:

```
fig.show()
```

- Gravar o código (por ex., 'figure.py') e executar para ver o resultado



The screenshot shows a window titled 'figure.py - C:\Users\Imrvr\_000\Desktop\Curso\_Python\_OBIO\exercícios\figure.py (3.7.1)'. The window contains a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Window', and 'Help'. The code editor displays the following Python code:

```
from matplotlib import pyplot as plt

#Cria uma figura vazia e exhibe
fig = plt.figure()
fig.suptitle("Sem título")
fig.show()
```

# Módulo pyplot

- Vamos agora adicionar dados à figura. Para começar, vamos introduzir os valores das contagens dos indivíduos em cada mês, sob a forma de uma lista de valores inteiros:  
ind = [5805, 4135, 1958, 788, 672, 627, 274, 730, 1335, 2624, 2592, 3835]
- Para cada contagem, é necessário indicar o mês do ano. Assim, vamos introduzir uma outra lista com os nomes dos meses do ano:  
meses = ["Jan", "Fev", "Mar", "Abr", "Mai", "Jun", "Jul", "Ago", "Set", "Out", "Nov", "Dez"]

```
*figure.py - C:\Users\Imrvr_000\Desktop\Curso_Python_OBIO\exercícios\figure.py (3.7.1)*
File Edit Format Run Options Window Help
from matplotlib import pyplot as plt

#Dados
ind = [5805, 4135, 1958, 788, 672, 627, 274, 730, 1335, 2624, 2592, 3835]
meses = ["Jan", "Fev", "Mar", "Abr", "Mai", "Jun", "Jul", "Ago", "Set", "Out",
        "Nov", "Dez"]

#Cria uma figura e exhibe
fig = plt.figure()
fig.suptitle("Sem título")
fig.show()
```

# Módulo pyplot

- Para adicionar estes dados à figura, usamos a função *bar()* para criar um gráfico de barras com o número de indivíduos em cada mês do ano:

`plt.bar (meses, ind) (*)`

```
figure.py - C:\Users\Imrvr_000\Desktop\Curso_Python_OBIO\exercícios\figure.py (3.7.1)
File Edit Format Run Options Window Help
from matplotlib import pyplot as plt

#Dados
ind = [5805, 4135, 1958, 788, 672, 627, 274, 730, 1335, 2624, 2592, 3835]
meses = ["Jan", "Fev", "Mar", "Abr", "Mai", "Jun", "Jul", "Ago", "Set", "Out",
        "Nov", "Dez"]

#Cria uma figura e exhibe
fig = plt.figure()
fig.suptitle("Sem título")
plt.bar(meses, ind)
fig.show()
```

\* Nota: O primeiro argumento da função são as categorias do eixo dos XX e o segundo argumento são os valores de Y



# Módulo pyplot

- Por defeito, a função `bar()` apresenta as barras verticais com a cor azul. Para obter outras cores, apenas tem de se acrescentar o argumento 'color' e o valor da cor respectiva como uma *string*. Testem as seguintes opções de cores e interpretem os respectivos resultados:

```
plt.bar(meses, ind, color = "r")
```

```
plt.bar(meses, ind, color = "m")
```

```
plt.bar(meses, ind, color = "g")
```

```
plt.bar(meses, ind, color = "c")
```

```
plt.bar(meses, ind, color = "br")
```

```
plt.bar(meses, ind, color = "bcy")
```



# Módulo pyplot

- Seguidamente, adicionamos o título dos eixos dos XX e dos YY e definimos o tamanho da respectiva letra, usando as funções `xlabel()` e `ylabel()`:

`plt.xlabel("meses", fontsize = 13)`

`plt.ylabel("número de indivíduos", fontsize = 13)`

```
figure.py - C:\Users\Imrvr_000\Desktop\Curso_Python_OBIO\exercícios\figure.py (3.7.1)
File Edit Format Run Options Window Help
from matplotlib import pyplot as plt

#Dados
ind = [5805, 4135, 1958, 788, 672, 627, 274, 730, 1335, 2624, 2592, 3835]
meses = ["Jan", "Fev", "Mar", "Abr", "Mai", "Jun", "Jul", "Ago", "Set", "Out",
         "Nov", "Dez"]

#Cria uma figura e exibe
fig = plt.figure()
fig.suptitle("Sem título")
plt.bar(meses, ind)
plt.xlabel("meses", fontsize = 13)
plt.ylabel("número de indivíduos", fontsize = 13)
fig.show()
```

# Módulo pyplot

- Podem adicionar-se linhas de grelha ao gráfico se for útil para visualizar facilmente os valores das barras. Neste caso, vamos adicionar linhas de grelha nos pontos principais do eixo dos YY, usando a função *grid()* com os seguintes argumentos:

`plt.grid (which = "major", axis = "y", color= "k", linestyle = "--", linewidth = 0.5)`

```
figure.py - C:\Users\Imrvr_000\Desktop\Curso_Python_OBIO\exercícios\figure.py (3.7.1)
File Edit Format Run Options Window Help
from matplotlib import pyplot as plt

#Dados
ind = [5805, 4135, 1958, 788, 672, 627, 274, 730, 1335, 2624, 2592, 3835]
meses = ["Jan", "Fev", "Mar", "Abr", "Mai", "Jun", "Jul", "Ago", "Set", "Out",
        "Nov", "Dez"]

#Cria uma figura e exhibe
fig = plt.figure()
fig.suptitle("Sem título")
plt.bar(meses, ind)
plt.xlabel("meses", fontsize = 13)
plt.ylabel("número de indivíduos", fontsize = 13)
plt.grid(which = "major", axis = "y", color='k', linestyle='--', linewidth=0.5)
fig.show()
```

# Módulo pyplot

- Além do gráfico de indivíduos/mês, temos também de elaborar um gráfico de doenças/mês. Uma opção seria criar um figura idêntica à anterior, mas podemos também usar a mesma figura e adicionar um outro gráfico. Para este fim, usamos a função *subplot()* do **pyplot**, que permite adicionar 2 ou mais gráficos a uma mesma figura
- Esta função tem como argumento um número inteiro de 3 dígitos (ou 3 números inteiros separados entre si) que determina a posição dos gráficos a incluir numa mesma figura:
  - Se os 3 inteiros são *nrows*, *ncols* e *index* por esta ordem, a função *subplot()* vai usar a posição do *index* numa grelha com linhas *nrows* e colunas *ncols*. O *index* começa em 1 no canto superior esquerdo e aumenta em direcção à direita
  - subplot (221) é equivalente a subplot (2,2,1)

# Módulo pyplot

- Para começar, temos de acrescentar ao nosso ficheiro de código uma nova lista com os totais do número de doenças por mês:

```
esp = [26, 23, 18, 8, 12, 8, 8, 7, 8, 15, 26, 26]
```

```
*figure.py - C:\Users\Imrvr_000\Desktop\Curso_Python_OBIO\exercícios\figure.py (3.7.1)*
File Edit Format Run Options Window Help
from matplotlib import pyplot as plt

#Dados
ind = [5805, 4135, 1958, 788, 672, 627, 274, 730, 1335, 2624, 2592, 3835]
meses = ["Jan", "Fev", "Mar", "Abr", "Mai", "Jun", "Jul", "Ago", "Set", "Out",
         "Nov", "Dez"]
esp = [26, 23, 18, 8, 12, 8, 8, 7, 8, 15, 26, 26]

#Cria uma figura e exhibe
fig = plt.figure()
fig.suptitle("Sem título")
plt.bar(meses, ind)
plt.xlabel("meses", fontsize = 13)
plt.ylabel("número de indivíduos", fontsize = 13)
plt.grid(which = "major", axis = "y", color='k', linestyle='--', linewidth=0.5)
fig.show()
```

# Módulo pyplot

- Por fim, vamos adicionar o título definitivo à figura e executar o código para ver o resultado final

```
figure.py - C:\Users\Imrvr_000\Desktop\Curso_Python_OBIO\exercícios\figure.py (3.7.1)
File Edit Format Run Options Window Help
from matplotlib import pyplot as plt

#Dados
ind = [5805, 4135, 1958, 788, 672, 627, 274, 730, 1335, 2624, 2592, 3835]
meses = ["Jan", "Fev", "Mar", "Abr", "Mai", "Jun", "Jul", "Ago", "Set", "Out",
        "Nov", "Dez"]
esp = [26, 23, 18, 8, 12, 8, 8, 7, 8, 15, 26, 26]

#Cria uma figura e exhibe
fig = plt.figure()
fig.suptitle("Variação mensal do número de indivíduos e doenças",
            fontsize = 18)

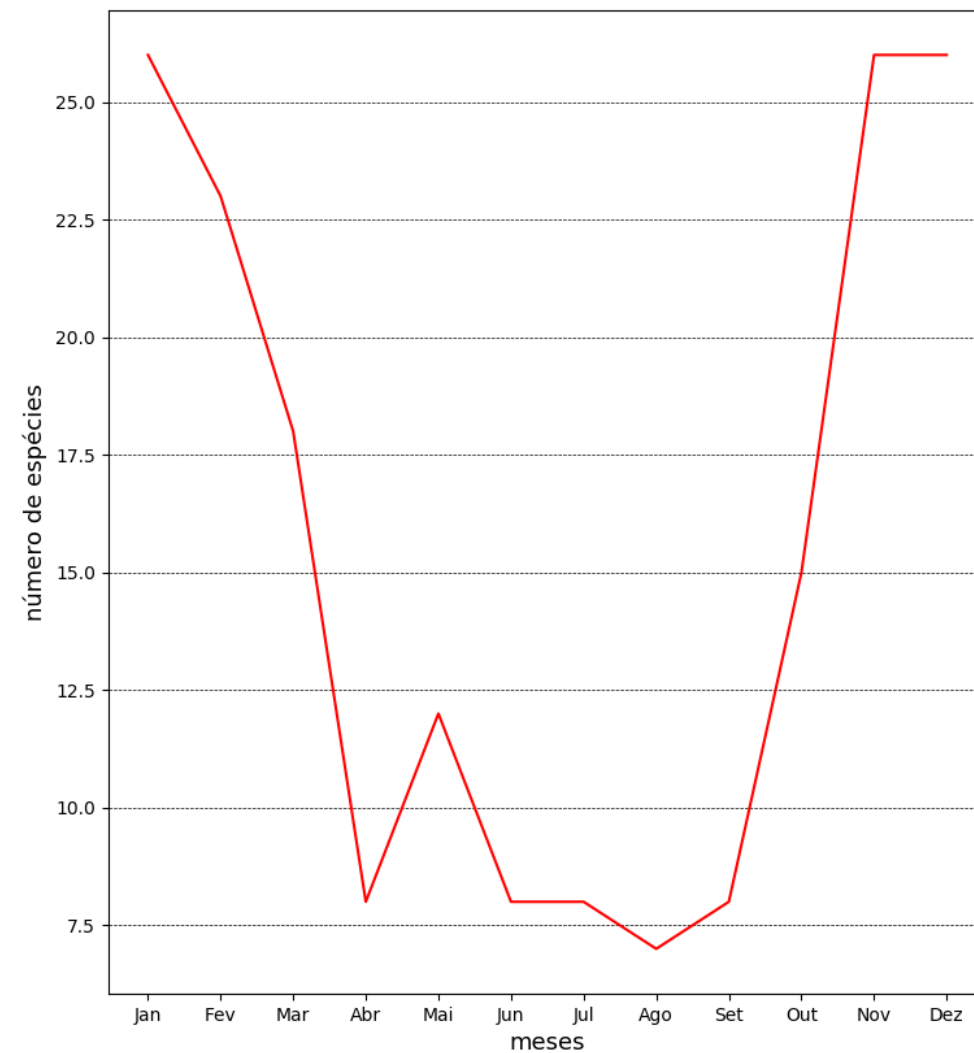
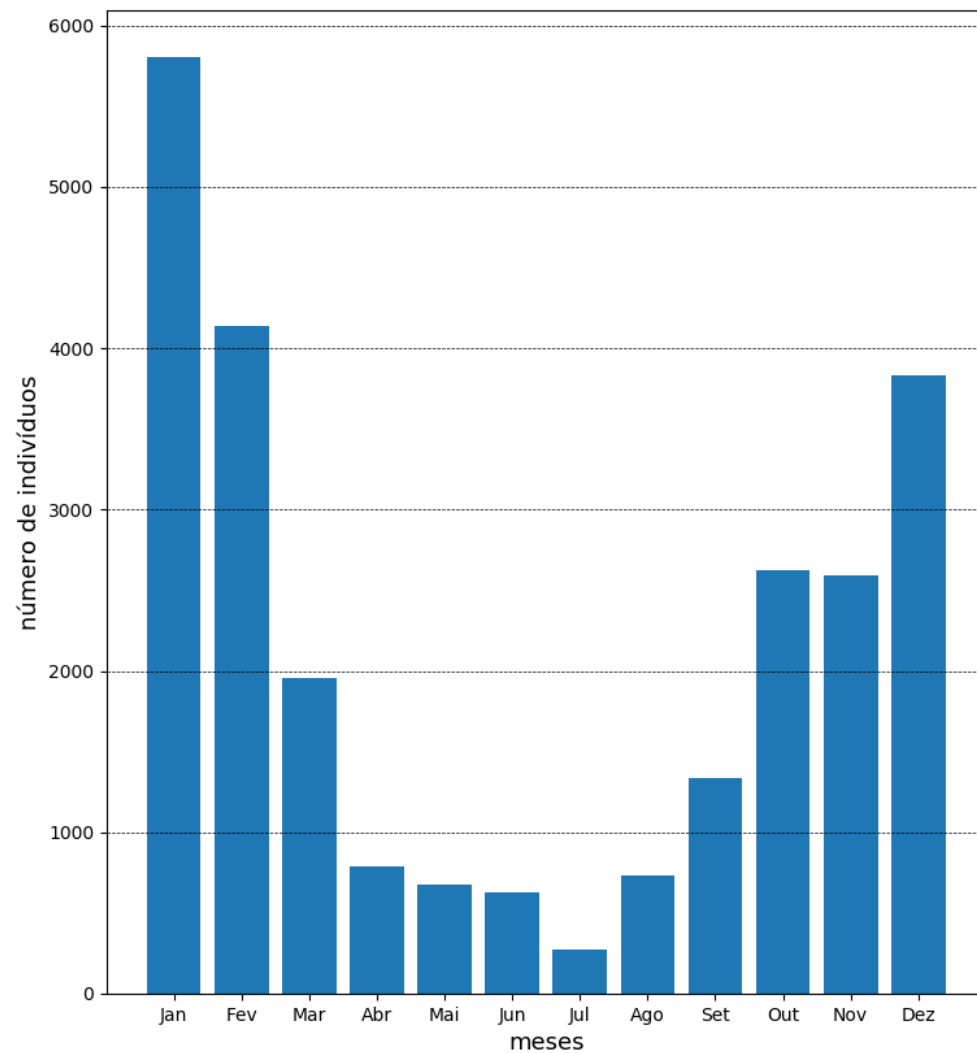
plt.subplot(121)
plt.bar(meses, ind)
plt.xlabel("meses", fontsize = 13)
plt.ylabel("número de indivíduos", fontsize = 13)
plt.grid(which = "major", axis = "y", color='k', linestyle='--', linewidth=0.5)

plt.subplot(122)
plt.plot(meses, esp, color = "r")
plt.xlabel("meses", fontsize = 13)
plt.ylabel("número de doenças", fontsize = 13)
plt.grid(which = "major", axis = "y", color='k', linestyle='--', linewidth=0.5)

fig.show()
```

# Módulo pyplot

Variação mensal do número de indivíduos e espécies no lago Taungthaman





# Módulo pyplot

- A figura pode ser gravada usando o ícone de gravação no canto inferior esquerdo ou, em alternativa, usando a instrução `fig.savefig("Figura 1")`

```
figure.py - C:\Users\Imrvr_000\Desktop\Curso_Python_OBIO\exercícios\figure.py (3.7.2)
File Edit Format Run Options Window Help
from matplotlib import pyplot as plt

#Dados
ind = [5805, 4135, 1958, 788, 672, 627, 274, 730, 1335, 2624, 2592, 3835]
meses = ["Jan", "Fev", "Mar", "Abr", "Mai", "Jun", "Jul", "Ago", "Set", "Out",
        "Nov", "Dez"]
esp = [26, 23, 18, 8, 12, 8, 8, 7, 8, 15, 26, 26]

#Cria uma figura e exhibe
fig = plt.figure(figsize = (17,11))
fig.suptitle("Variação mensal do número de indivíduos e doenças",
            fontsize = 18)

plt.subplot(121)
plt.bar(meses, ind)
plt.xlabel("meses", fontsize = 13)
plt.ylabel("número de indivíduos", fontsize = 13)
plt.grid(which = "major", axis = "y", color='k', linestyle='--', linewidth=0.5)

plt.subplot(122)
plt.plot(meses, esp, color = "r")
plt.xlabel("meses", fontsize = 13)
plt.ylabel("número de doenças ", fontsize = 13)
plt.grid(which = "major", axis = "y", color='k', linestyle='--', linewidth=0.5)

fig.show()
fig.savefig("Figura 1")
```



# Exercícios

- Uma equipa de biólogos realizou uma expedição às florestas da Costa Rica para estudar 2 géneros de preguiças, *Bradypus* e *Choloepus*. O objectivo era determinar se o comprimento do pelo das preguiças está relacionado com o número de outros organismos de diferentes grupos taxonómicos (por ex., algas ou insectos), que se escondem no interior do pelo daqueles animais. Para tal, os investigadores mediram o comprimento do pelo numa área específica do corpo das preguiças e contaram o número de organismos escondidos nessa área do pelo. Os dados foram registados numa tabela (“preguiças.csv”) para cada um dos animais estudados



# Exercícios

- O nº total de organismos presentes no pelo das preguiças dos 2 géneros pode ser extraído do ficheiro csv usando a função *preguicas* do ficheiro *preguiça.py*. Para perceber se há uma associação entre o comprimento do pelo e o número de organismos é necessário construir um gráfico de linha para cada género de preguiças. Para este fim, deverão efectuar o seguinte:
  1. Definir uma função que permita gerar um gráfico de linha para cada género de preguiça, que relacione o comprimento do pelo com o número de organismos
  2. Interpretar o código da função *preguicas* e fazer as alterações necessárias por forma a que a função do gráfico seja invocada no corpo daquela função



# Fim da sessão 5

