Aula 5 - Instalando e carregando m?dulos

Karl Jan Clinckspoor

2 de julho de 2018

Sumário

1 Introdução

Um dos grandes pontos positivos do Python é sua biblioteca interna bastante completa e extenso ecosistema de pacotes. É difícil pensar num pacote que não foi feito, e se não foi, qualquer um pode publicar um pacote, desde que ele se atenha a alguns detalhes. Além disso, é muito fácil instalar pacotes. Isso é feito de uma maneira principal (pip), e de outra complementar (conda).

1.1 *pip*

O método principal de instalação de pacotes é pelo *pip*, que significa *pip installs packages*. A sintaxe de instalação é:

```
pip install <nome do pacote>
```

Porém, isso deve ser feito não no Jupyter Notebook, mas no console. Aperte a tecla do windows + R e digite 'cmd' ou 'powershell', sem as aspas, e digite pip. Se o seguinte aparecer, tudo está OK para continuar.

```
Usage:
  pip <command> [options]

Commands:
```

```
  install                     Install packages.
  download                    Download packages.
  uninstall                   Uninstall packages.
  freeze                      Output installed packages in requirements format.
  list                        List installed packages.
  show                        Show information about installed packages.
[...]
```

Se ele retornar um erro do tipo *pip não foi encontrado*, significa que *pip* não está na variável PATH de ambiente. Para fazer isso, vá até o painel de controle, clique em sistema, configurações avançadas, aba avançadas, variáveis de ambiente. Na parte debaixo, em variáveis de sistema, clique em PATH, editar, e adicione o caminho de onde tanto *pip* quanto *python* estão. Para encontrar isso, encontre esses arquivos pela busca do sistema, e pegue o caminho da barra de endereço.

Caso você tenha instalado o anaconda ou o miniconda, é possível instalar pacotes de outra maneira.

1.2 *conda*

Conda instala qualquer pacote em ambientes conda, e pip instala pacotes do Python em qualquer ambiente. Isso pode não fazer muito sentido, mas basicamente, se você quiser instalar um novo pacote e você tem Anaconda instalado, utilize conda, possivelmente com a opção conda-forge. Senão, use pip. Aqui, os pacotes que serão instalados são bastante simples, e o pip é o suficiente.

2 Instalação de pacotes pelo console

Abra uma janela do console, ou utilize a janela já aberta com o pip. Digite:

```
pip install uncertainties sympy
```

Isso irá instalar o pacote para propagação de incertezas e um pacote para matemática simbólica. Caso você não consiga rodar o pip, tente rodar o seguinte (copie e cole numa cela -- medida de proteção contra execução indesejada do código).

```
import sys
!{sys.executable} -m pip install uncertainties sympy
```

Para encontrar pacotes, basta utilizar o google ou procurar no Python Package Index, PyPI
Link muito útil discutindo as diferenças entre pip e conda

3 Utilização de pacotes - *import*

Para começar a utilizar um pacote, é necessário o comando **import**. Há várias maneiras que isso pode ser feito.

1. import {pacote}
2. import {pacote} as {apelido}
3. from {pacote} import {parte}

O primeiro método importa o pacote normalmente. Porém, para utilizar o conteúdo do pacote, é necessário utilizar a notação de ponto. Caso o nome do pacote seja muito comprido, isso fica bastante inconveniente. Logo, é frequente utilizar apelidos para os pacotes (*alias*). Caso você deseje utilizar as funções de um pacote sem ter que preceder as funções com o nome do pacote, pode utilizar a terceira notação. Caso deseje importar tudo, pode utilizar **from {pacote} import *** , mas isso não é muito recomendado. Veja os seguintes exemplos.

```
In [2]: from uncertainties import ufloat
        import sympy as sp
```

Caso você tenha instalado ambos os pacotes corretamente, essa etapa irá ocorrer sem maiores problemas.

4 Uncertainties

Esse pacote facilita muito a propagação de erro, mesmo de equações complexas. É necessário somente criar as variáveis com os valores e erros associados, e depois realizar as contas como normal. No final, o valor terá seu erro propagado. O pacote não é perfeito, e não funciona muito bem para coisas muito complexas (veja a documentação do pacote para mais detalhes, aqui).

Veja o exemplo, utilizando *r* como o raio de uma esfera e um valor aproximado para pi.

```
In [5]: r = ufloat(13, 1)
        pi = ufloat(3.14, 0.01)

        V = 4/3 * pi * r ** 3
        A = 4 * pi * r ** 2
        p = 2 * pi * r

        print(f'O raio da esfera é {r}, seu volume é {V}, sua área é {A} e o perímetro de um cír

O raio da esfera é 13.0+/-1.0, seu volume é (9.2+/-2.1)e+03, sua área é (2.12+/-0.33)e+03 e o pe
```

Calcular coisas mais complexas fica a critério de sua criatividade.

5 Sympy

O pacote sympy (homepage) é utilizado para fazer cálculos simbólicos. Por exemplo, é possível fornecer uma equação e obter sua derivada parcial em relação a um termo. Além disso, ele consegue mostrar as equações de uma maneira bastante bonita. Vamos começar definindo uma função simples, e depois realizando algumas operações simples

```
In [10]: sp.init_printing(use_latex=True) # Caso isso não funcione, utilize use use_unicode=True

         x, y, a, b, c = sp.symbols('x y a b c')
         f = a * x ** 2 + b * x + c
         f
```

Out[10]:

$$ax^2 + bx + c$$

In [12]:
```
sp.Derivative(f, x)  # Mostra a operação de derivação
```

Out[12]:

$$\frac{\partial}{\partial x}\left(ax^2 + bx + c\right)$$

In [14]:
```
sp.derive_by_array(f, x)  # Executa a derivação
```

Out[14]:

$$2ax + b$$

In [16]:
```
sp.Integral(f, x)  # Mostra a operação de integração
```

Out[16]:

$$\int \left(ax^2 + bx + c\right)\, dx$$

In [18]:
```
sp.integrate(f, x)  # Integra (não se esqueça do C!)
```

Out[18]:

$$\frac{ax^3}{3} + \frac{bx^2}{2} + cx$$

In [23]:
```
f2 = sp.Equality(y, f) # Cria uma igualdade
f2
```

Out[23]:

$$y = ax^2 + bx + c$$

In [24]:
```
sp.derive_by_array(f2, x)  # Deriva a igualdade, mostrando a derivação parcial
```

Out[24]:

$$\frac{\partial}{\partial x} y = ax^2 + bx + c$$

In [27]:
```
G0, tr, w = sp.symbols('G_0, tau_r, omega')  # Define os símbolos do modelo de Maxwell
```

In [31]:
```
G1 = G0 * (tr * w) ** 2 / ((tr * w) ** 2 + 1)  # G'
G2 = G0 * (tr * w) / ((tr * w) ** 2 + 1)       # G''
```

In [32]:
```
G1
```

Out[32]:

$$\frac{G_0\omega^2\tau_r^2}{\omega^2\tau_r^2+1}$$

In [34]: G2

Out[34]:

$$\frac{G_0\omega\tau_r}{\omega^2\tau_r^2+1}$$

In [54]:
```
der1 = sp.derive_by_array(G1, w)
der2 = sp.derive_by_array(G2, w)  # Derivadas
der1
```

Out[54]:

$$-\frac{2G_0\omega^3\tau_r^4}{\left(\omega^2\tau_r^2+1\right)^2}+\frac{2G_0\omega\tau_r^2}{\omega^2\tau_r^2+1}$$

In [55]: der2

Out[55]:

$$-\frac{2G_0\omega^2\tau_r^3}{\left(\omega^2\tau_r^2+1\right)^2}+\frac{G_0\tau_r}{\omega^2\tau_r^2+1}$$

In [52]: sp.solveset(der2, w)

Out[52]:

$$\left\{-\frac{1}{\tau_r},\frac{1}{\tau_r}\right\}\setminus\left\{-\frac{i}{\tau_r},\frac{i}{\tau_r}\right\}$$

In [56]: sp.solveset(G1 - G2, w) # Resolução de quando G' = G'' (regime linear)

Out[56]:

$$\left\{0,\frac{1}{\tau_r}\right\}\setminus\left\{-\frac{i}{\tau_r},\frac{i}{\tau_r}\right\}$$

Como pode ser visto, o sympy é uma ferramenta bastante poderosa. Brinque como desejar com ela.

6 *glob, os*

Esses dois pacotes vem inclusos com o Python.

A função do glob é produzir listas que contém nomes de arquivos seguindo uma determinada regra. Ele é muito utilizado para, por exemplo, criar uma lista com todos os nomes dos experimentos que serão tratados.

A função de os é providenciar funções gerais para operações com o sistema operacional, seja ele Windows, Linux ou Mac. Esse pacote pode ser utilizado para renomear arquivos e também para alterar o diretório ativo (onde o Python procura primeiro por arquivos).

Neste exemplo, iremos alterar o diretório atual para o diretório que contém os arquivos, e depois criar uma lista com seus nomes. Aí, os arquivos serão renomeados.

```
In [1]: import os
        import glob

In [16]: print(os.getcwd())
         os.chdir(r'./dados-1')  # '.' significa o diretório atual, então isso significa a pasta
         print(os.getcwd())

/home/karl/dados/Dropbox/Python/Curso
/home/karl/dados/Dropbox/Python/Curso/dados-1
```

Agora que mudamos de diretório, conseguimos acessar facilmente os arquivos. Para isso, vamos utilizar o glob.

```
In [17]: arquivos = glob.glob('*.dat')
         arquivos

Out[17]: ['39.dat',
          '41.dat',
          '43.dat',
          '38.dat',
          '37.dat',
          '45.dat',
          '40.dat',
          '42.dat',
          '44.dat']
```

Como podemos ver, há 9 arquivos com alguns nomes pouco descritivos. Vamos agora melhorar a descrição deles. Isso é útil porque, ao tratar um dos arquivos, podemos nos referir ao nome dele para, por exemplo, colocar na legenda de uma figura, sem ter que configurar isso separadamente.

Primeiro começamos definindo os novos nomes e checamos se tudo está certo.

```
In [19]: novos_nomes = ['SAXS_CTAB_Ur' + i for i in arquivos]
         novos_nomes
```

```
Out[19]: ['SAXS_CTAB_Ur39.dat',
          'SAXS_CTAB_Ur41.dat',
          'SAXS_CTAB_Ur43.dat',
          'SAXS_CTAB_Ur38.dat',
          'SAXS_CTAB_Ur37.dat',
          'SAXS_CTAB_Ur45.dat',
          'SAXS_CTAB_Ur40.dat',
          'SAXS_CTAB_Ur42.dat',
          'SAXS_CTAB_Ur44.dat']
```

Agora alteramos os nomes dos arquivos para esses novos nomes.

```
In [20]: for velho, novo in zip(arquivos, novos_nomes):
             os.rename(velho, novo)

In [21]: glob.glob('*.dat')

Out[21]: ['SAXS_CTAB_Ur38.dat',
          'SAXS_CTAB_Ur43.dat',
          'SAXS_CTAB_Ur45.dat',
          'SAXS_CTAB_Ur41.dat',
          'SAXS_CTAB_Ur39.dat',
          'SAXS_CTAB_Ur42.dat',
          'SAXS_CTAB_Ur44.dat',
          'SAXS_CTAB_Ur37.dat',
          'SAXS_CTAB_Ur40.dat']
```

Para resetar os nomes ao formato original, rode o seguinte comando que é uma inversão do comando anterior.

```
In [22]: for velho, novo in zip(novos_nomes, arquivos):
             os.rename(velho, novo)
```

Espero que consiga ver o quão útil isso pode ser. Imagine que você receba uma grande quantidade de arquivos com nomes irregulares, alguns maiúsculos, outros minúsculos, com erros de digitação nos compostos, etc. Seria muito chato alterar os nomes manualmente. Com essas ferramentas, isso se torna muito mais fácil.

Não é necessário rodar isso por um Jupyter Notebook. Você pode abrir uma seção de Python interativo pelo console e copiar e colar os códigos das celas anteriores, alterando de acordo com sua necessidade.