Lista.Magics

October 27, 2019

[1]:
```
%cd ..
```

```
/home/joao/projects/tutorial/pdf/source
```

```
<IPython.core.display.Javascript object>
```

1 Lista Magics do IPython

Podemos usar a magic `%lsmagic` para listar quais são todas as magics do IPython, a magic `%magic` para entender como funciona a parte de magics, a magic `%quickref` para ter uma referência rápida de que extensões ao Python o IPython faz.

[2]:
```
%lsmagic
```

[2]:
```
Available line magics:
%alias  %alias_magic  %autoawait  %autocall  %automagic  %autosave  %bookmark
%cat  %cd  %clear  %colors  %conda  %config  %connect_info  %cp  %debug  %dhist
%dirs  %doctest_mode  %ed  %edit  %env  %gui  %hist  %history  %killbgscripts
%ldir  %less  %lf  %lk  %ll  %load  %load_ext  %loadpy  %logoff  %logon
%logstart  %logstate  %logstop  %ls  %lsmagic  %lx  %macro  %magic  %man
%matplotlib  %mkdir  %more  %mv  %notebook  %page  %pastebin  %pdb  %pdef  %pdoc
%pfile  %pinfo  %pinfo2  %pip  %popd  %pprint  %precision  %prun  %psearch
%psource  %pushd  %pwd  %pycat  %pylab  %qtconsole  %quickref  %recall  %rehashx
%reload_ext  %rep  %rerun  %reset  %reset_selective  %rm  %rmdir  %run  %save
%sc  %set_env  %store  %sx  %system  %tb  %time  %timeit  %unalias  %unload_ext
%who  %who_ls  %whos  %xdel  %xmode

Available cell magics:
%%!  %%HTML  %%SVG  %%bash  %%capture  %%debug  %%file  %%html  %%javascript
%%js  %%latex  %%markdown  %%perl  %%prun  %%pypy  %%python  %%python2
%%python3  %%ruby  %%script  %%sh  %%svg  %%sx  %%system  %%time  %%timeit
%%writefile

Automagic is ON, % prefix IS NOT needed for line magics.
```

Alias

- `%alias`: define novas magics baseadas em comandos do sistema
- `%alias_magic`: define novas magics baseadas em magics existentes
- `%unalias`: remove alias

É possível criar um alias para a mágic já com os argumentos:

[3]:
```
%alias_magic ls_int who_ls -p "int"
%ls_int
```

```
Created `%ls_int` as an alias for `%who_ls int`.
```

[3]:
```
[]
```

Histórico

- `%load`: carrega um script externo em uma célula. É possível especificar quais linhas, classes e funções do script devem ser carregadas. Também é possível passar números de execução como argumento para re-executar células anteriores e macros.
- `%loadpy`: quase um alias de `%load`, mas que carrega arquivos Python sem o '.py'
- `%history`: exibe histórico de execução, como mostramos antes
- `%hist`: alias para `%history`
- `%macro`: permite definir um nome para parte do histórico de execução e usar esse nome em `%load`
- `%notebook`: exporta histórico para um novo arquivo de notebook
- `%save`: salva macro em arquivo

[ ]:
```
# %load 1-2
%cd ..
%reload_ext slide
%lsmagic
```

Extensões

- `%load_ext`: importa uma extensão do IPython. Funciona como um import normal seguindo de uma chamada a função `load_ipython_extension(shell)` definida no módulo. É possível usar isso para definir módulos com magics, como veremos posteriormente
- `%unload_ext`: desabilita uma extensão do IPython ao chamar `unload_ipython_extension(shell)`
- `%reload_ext`: executa as duas magics anteriores

Namespace

- `%psearch`: busca nome por algum padrão
- `%who_ls`: retorna nomes de determinado tipo
- `%who`: semelhante a `%who_ls`, mas apenas exibe nomes
- `%whos`: exibe tabela com variável, tipo e valor

- `%rehashx`: atualiza a tabela de alias para adicionar todos os executáveis de $PATH
- `%reset`: recarrega o namespace
- `%reset_selective`: remove nomes definidos pelo usuário
- `%xdel`: remove variável e tenta limpar tudo do IPython que usa ela

Variáveis de ambiente

- `%env`: lista, define e lê variáveis de ambiente
- `%set_env`: mesmo que `%env`, mas sem a parte da leitura

[5]:
```
%env PATH
```

[5]:
```
'/home/joao/anaconda3/bin:/home/joao/anaconda3/condabin:/home/joao/.rvm/gems/rub
y-2.4.0/bin:/home/joao/.rvm/gems/ruby-2.4.0@global/bin:/home/joao/.rvm/rubies/ru
by-2.4.0/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/g
ames:/usr/local/games:/home/joao/.rvm/bin:/home/joao/.rvm/bin:/home/joao/.rvm/bi
n'
```

[6]:
```
number = 1
%env NOTEBOOK_NUMBER=$number
```

```
env: NOTEBOOK_NUMBER=1
```

Configurações Habilita, desabilita e configura funcionalidades do IPython:

- `%autoawait`: chama await de corotinas automaticamente
- `%autocall`: executa funções sem usar parênteses
- `%automagic`: executa magics sem usar `%`
- `%autosave`: define o tempo para salvar o notebook automaticamente
- `%config`: permite configurar outros aspectos do IPython
- `%pprint`: habilita/desabilita visualização "pretty"
- `%precision`: define a precisão do float para o pprint
- `%matplotlib`: permite o uso interativo do matplotlib
- `%xmode`: altera o modo de tratamento de exceções

Visualização

- `%%HTML`, `%%html`: renderiza célula como html
- `%%SVG`, `%%svg`: renderiza célula como svg
- `%%javascript`, `%%js`: renderiza célula como javascript/executa no navegador
- `%%latex`: renderiza célula como latex
- `%%markdown`: renderiza célula como markdown

Diretórios

- `%cd`: altera o diretório de execução do notebook
- `%bookmark`: salva diretórios para facilitar o uso de `cd`

- `%dhist`: exibe o histórico de diretórios navegados
- `%dirs`: exibe a pilha de diretórios
- `%popd`: retira diretório da pilha
- `%pushd`: insere diretório na pilha
- `%pwd`: retorna diretório atual

Log

- `%logstart`: cria um arquivo de log para armazenar todas as células executadas no Jupyter
- `%logstop`: fecha o arquivo de log
- `%logoff`: pausa o log temporariamente, mas não fecha o arquivo de log
- `%logon`: volta a fazer log das operações
- `%logstate`: exibe o estado do log

Documentação Magics relacionadas a documentação

- `%page`: exibe objeto no pager
- `%pdef`: exibe assinatura de função
- `%pdoc`: exibe docstring de objeto
- `%pfile`: exibe arquivo de definição de objeto
- `%pinfo`: exibe documentação (equivalente a `?`)
- `%pinfo2`: exibe código fonte (equivalente a `??`)
- `%psource`: exibe o código fonte de objeto no pager
- `%pycat`: exibe o código fonte de objeto no pager com syntax-highlighting

Debug e profile

- `%debug`: ativa debug interativo
- `%pdb`: configura o uso do debug interativo
- `%prun`: executa comando ou expressão com profiler
- `%%prun`: executa célula com profiler
- `%tb`: exibe último traceback
- `%time`: calcula tempo de execução de expressão ou comando
- `%timeit`: calcula média de tempo de execução de expressão ou comando ao executar várias vezes
- `%%time`: calcula tempo de execução de célula
- `%%timeit`: calcula média de tempo de execução de célula ao executar várias vezes

IPython no console

- `%colors`: muda o esquema de cores
- `%doctest_mode`: altera o modo de execução para ficar parecido com o shell clássico do IPython
- `%ed`, `%edit`: abre editor de texto
- `%gui`: define a forma de exibição (qt, gtk, tk, wx, …)
- `%recall`, `%rep`: carrega linhas do histórico na próxima linha do prompt
- `%rerun`: re-executa input anterior

Execução externa

- `%run`: executa arquivo no IPython como um programa
- `%sc`: alternativa descontinuada de bang expression (`!`)
- `%sx`, `%system`: alternativa a bang expression que retorna lista (`!!`)
- `%%!`, `%%sx`, `%%system`: executa célula com comandos do sistema
- `%%bash`: executa célula com bash em um subprocesso
- `%%sh`: executa célula com sh em um subprocesso
- `%%perl`: executa célula com perl em um subprocesso
- `%%pypy`: executa célula com pypy em um subprocesso
- `%%python`: executa célula com python em um subprocesso
- `%%python2`: executa célula com python2 em um subprocesso
- `%%python3`: executa célula com python3 em um subprocesso
- `%%ruby`: executa célula com ruby em um subprocesso
- `%%script`: executa célula com interpretador externo em um subprocesso (especifique o interpretador com o parâmetro)

Alias para comandos do sistema

- `%cat`
- `%clear`
- `%cp`
- `%ldir`
- `%less`
- `%lf`
- `%lk`
- `%ll`
- `%ls`
- `%lx`
- `%man`
- `%mkdir`
- `%more`
- `%mv`
- `%rm`
- `%rmdir`

Outros

- `%conda`: executa conda no kernel atual
- `%pip`: executa pip no kernel atual
- `%pylab`: carrega `matplotlib` e `numpy` e expõe diversos métodos no escopo global
- `%killbgscripts`: mata scripts do plano de fundo iniciados pelo notebook
- `%pastebin`: envia código para o pastebin. Aceita range do histórico, nome do arquivo, macro, ou string
- `%qtconsole`: abre um qtconsole conectado a este kernel
- `%store`: salva variáveis Python para usar entre sessões
- `%connect_info`: exibe informações de conexão do kernel para que outros clientes possam acessar o mesmo kernel

- `%%capture`: executa célula capturando output
- `%%file`, `%%writefile`: escreve célula em arquivo