Python notes

Contents

Python **1**
Basic packages 1
Coding style PEP 8 summary 1

Spyder IDE **2**
Shortcuts 2

IPython **2**
Special commands 2
IPython's QT console 3
Debugger in IPython 3
Graphics 3

Python

Basic packages

- Python home page (http://www.python.org)
- **Pylab/Matplotlib** (plotting as in Matlab)
- **Numpy** (fast vectors and matrices, (NUMerical PYthon)
- **SciPy** (scientific algorithms, odeint)
- **Visual Python** (3d visualisation)
- **SymPy** (Symbolic calculation)

Coding style PEP 8 summary

- Indentation: use 4 spaces
- One space around assignment operator (=) operator: `c = 5` and not `c=5`.
- Spaces around arithmetic operators can vary: `x = 3*a + 4*b` is okay, but also okay to write `x = 3 * a + 4 * b`.
- No space before and after parentheses: `x = sin(x)` but not `x = sin( x )`
- A space after comma: `range(5, 10)` and not `range(5,10)`.
- No whitespace at end of line
- No whitespace in empty line
- One or no empty line between statements within function
- Two empty lines between functions
- One import statement per line
- import first standand Python library (such as `math`), then third-party packages (`numpy`, `scipy`, ...), then our own modules
- no spaces around `=` when used in keyword arguments `"Hello World".split(sep=' ')` but not `"Hello World".split(sep = ' ')`.

`pep8()` program available to check source code from command line.

Spyder IDE

Shortcuts

Ctrl+Enter executes the current cell (menu entry Run > Run cell). A cell is defined as the code between two lines which start with the agreed tag `#%%`.

Shift+Enter executes the current cell and advances the cursor to the next cell (menu entry Run > Run cell and advance).

Cells are useful to execute a large file/code segment in smaller units. (It is a little bit like a cell in an IPython notebook, in that chunks of code can be run independently.)

Alt+<Up Arrow> moves the current line up. If multiple lines are highlighted, they are moved up together.

Alt+<Down arrow> works correspondingly moving line(s) down.

Ctrl+Left Mouse Click on a function/method in the source, opens a new editor windows showing the definition of that function.

Shift+Ctrl+Alt+M maximizes the current window (or changes the size back to normal if pressed in a maximized window)

Ctrl+Shift+F activates the search pane across all files.

Cmd + + (On MacOS X) or

Ctrl + + (otherwise) will increase the font size in the Editor, whereas

Cmd + - (Ctrl + -) will decrease it. Also works in the IPython Console.

The font size for the Help, the Python console etc. can be set individually via Preferences > Help etc.

I couldn't find a way of changing the font size in the variable explorer.

Cmd+S (on MacOS X) and

Ctrl+S (otherwise) in the Editor pane saves the file currently being edited. This also forces various warning triangles in the left column of the Editor to be updated (otherwise they update every 2 to 3 seconds by default).

Cmd+S (on MacOS X) and

Ctrl+S (otherwise) in the IPython console pane saves the current IPython session as an HTML file, including any figures that may be displayed inline. This is useful as a quick way of recording what has been done in a session.

(It is not possible to load this saved record back into the session - if you need functionality like this, look for the IPython Notebook.)

Cmd+I (on Mac OS X) and

Ctrl+I (otherwise) when pressed while the cursor is on an object, opens documentation for that object in the help pane.

IPython

Special commands

Interactive Python (`ipython` from Command Prompt/Unix-shell)

- command history (across sessions), auto completion,
- special commands:
  - `%run` test will execute file `test.py` in current name space (in contrast to IDLE this does not remove all existing objects from global name space)
  - `%reset` can delete all objects if required
  - use `range?` instead of `help(range)`
  - `%logstart` will log your session

- `%prun` will profile code
- `%timeit` can measure execution time
- `%load` loads file for editing
- Much (!) more (read at http://ipython.org)

IPython's QT console

- Prompt as IPython (with all it's features): running in a graphics console rather than in text console
- can inline `matplotlib` figures
- Read more at http://ipython.org/ipythondoc/dev/interactive/qtconsole.html

Debugger in IPython

You can also control the debugging process by issuing these commands in the console prompt:

`n` to move to the Next statement.

`s` to Step into the current statement. If this is a function call, step into that function.

`r` to complete all statements in the current function and Return from that function before returning control.

`p` to print values of variables, for example `p x` will print the value of the variable `x`.

If you use `%debug`, you may also want to use the commands `up` (i.e. press `u` at the debugger) and `down` (i.e. press `d`) which navigate the inspection point up and down the stack. (Up the stack means to the functions that have called the current function; down is the opposite direction.)

Graphics

The command to get the figures to appear inline in the IPython console is:

```
In [3]: %matplotlib inline
```

The command to get figures appear in their own window (which technically is a Qt window) is:

```
In [4]: %matplotlib qt
```

The Spyder preferences can be used to customize the default behavior (in particular `Preferences > IPython Console > Graphics > Activate Support` to switch into inline plotting).