

How to access Python for doing scientific computing

Hans Petter Langtangen^{1,2}

¹Center for Biomedical Computing, Simula Research Laboratory

²Department of Informatics, University of Oslo

Mar 23, 2015

A comprehensive eco system for scientific computing with Python used to be quite a challenge to install on a computer, especially for newcomers. This problem is more or less solved today. There are several options for getting easy access to Python and the most important packages for scientific computations, so the biggest issue for a newcomer is to make a proper choice. An overview of the possibilities together with my own recommendations appears next.

1 Required software

The strictly required software packages for working with this book are

- Python¹ version 2.7 [11]
- Numerical Python² (NumPy) [8, 7] for array computing
- Matplotlib³ [4, 3] for plotting

Desired add-on packages are

- IPython⁴ [10, 9] for interactive computing
- SciTools⁵ [6] for add-ons to NumPy
- ScientificPython⁶ [2] for add-ons to NumPy

¹<http://python.org>

²<http://www.numpy.org>

³<http://matplotlib.org>

⁴<http://ipython.org>

⁵<https://github.com/hplgit/scitools>

⁶<http://starship.python.net/crew/hinzen>

- `pytest`⁷ or `nose`⁸ for testing programs
- `pip`⁹ for installing Python packages
- `Cython`¹⁰ for compiling Python to C
- `SymPy`¹¹ [1] for symbolic mathematics
- `SciPy`¹² [5] for advanced scientific computing

There are different ways to get access to Python with the required packages:

1. Use a computer system at an institution where the software is installed. Such a system can also be used from your local laptop through remote login over a network.
2. Install the software on your own laptop.
3. Use a web service.

A system administrator can take the list of software packages and install the missing ones on a computer system. For the two other options, detailed descriptions are given below.

Using a web service is very straightforward, but has the disadvantage that you are constrained by the packages that are allowed to install on the service. There are services at the time of this writing that suffice for working with most of this book, but if you are going to solve more complicated mathematical problems, you will need more sophisticated mathematical Python packages, more storage and more computer resources, and then you will benefit greatly from having Python installed on your own computer.

This author's experience is that installation of mathematical software on personal computers quickly becomes a technical challenge. Linux Ubuntu (or any Debian-based Linux version) contains the largest repository today of pre-built mathematical software and makes the installation trivial without any need for particular competence. Despite the user-friendliness of the Mac and Windows systems, getting sophisticated mathematical software to work on these platforms requires considerable competence.

2 Installing software on your laptop: Mac OS X and Windows

There are various possibilities for installing the software on a Mac OS X or Windows platform:

⁷<http://pytest.org/latest/>

⁸<https://nose.readthedocs.org>

⁹<http://www.pip-installer.org>

¹⁰<http://cython.org>

¹¹<http://sympy.org>

¹²<http://scipy.org>

1. Use `.dmg` (Mac) or `.exe` (Windows) files to install individual packages
2. Use Homebrew or MacPorts to install packages (Mac only)
3. Use a pre-built rich environment for scientific computing in Python:
 - Anaconda¹³
 - Enthought Canopy¹⁴
4. Use a virtual machine running Ubuntu:
 - VMWare Fusion¹⁵
 - VirtualBox¹⁶
 - Vagrant¹⁷

Alternative 1 is the obvious and perhaps simplest approach, but usually requires quite some competence about the operating system as a long-term solution when you need many more Python packages than the basic three. This author is not particularly enthusiastic about Alternative 2. If you anticipate to use Python extensively in your work, I strongly recommend operating Python on an Ubuntu platform and going for Alternative 4 because that is the easiest and most flexible way to build and maintain your own software ecosystem. Alternative 3 is recommended for those who are uncertain about the future needs for Python and think Alternative 4 is too complicated. My preference is to use Anaconda for the Python installation and Spyder¹⁸ (comes with Anaconda) as a graphical interface with editor, an output area, and flexible ways of running Python programs.

3 Anaconda and Spyder

Anaconda¹⁹ is a free Python distribution produced by Continuum Analytics and contains about 200 Python packages, as well as Python itself, for doing a wide range of scientific computations. Anaconda can be downloaded from <http://continuum.io/downloads>. Choose Python version 2.7.

The Integrated Development Environment (IDE) Spyder is included with Anaconda and is my recommended tool for writing and running Python programs on Mac and Windows, unless you have preference for a plain text editor for writing programs and a terminal window for running them.

¹³<https://store.continuum.io/cshop/anaconda/>

¹⁴<https://www.enthought.com/products/canopy/>

¹⁵<http://www.vmware.com/products/fusion>

¹⁶<https://www.virtualbox.org/>

¹⁷<http://www.vagrantup.com/>

¹⁸<https://code.google.com/p/spyderlib/>

¹⁹<https://store.continuum.io/cshop/anaconda/>

3.1 Spyder on Mac

Spyder is started by typing `spyder` in a (new) Terminal application. If you get an error message *unknown locale*, you need to type the following line in the Terminal application, or preferably put the line in your `$HOME/.bashrc` Unix initialization file:

```
export LANG=en_US.UTF-8; export LC_ALL=en_US.UTF-8
```

3.2 Installation of additional packages

Anaconda installs the `pip` tool that is handy to install additional packages. In a Terminal application on Mac or in a PowerShell terminal on Windows, write

Terminal

```
pip install --user packagename
```

3.3 Installing SciTools on Mac

The SciTools package can be installed by `pip` on Mac:

Terminal

```
Terminal> pip install --user -e \
            git+https://github.com/hplgit/scitools.git#egg=scitools
```

3.4 Installing SciTools on Windows

The safest procedure on Windows is to go to <https://github.com/hplgit/scitools/> and click on *Download ZIP*. Double-click on the downloaded file in Windows Explorer and perform *Extract all files* to create a new folder with all the SciTools files. Find the location of this folder, open a PowerShell window, and move to the location, e.g.,

Terminal

```
Terminal> cd C:\Users\username\Downloads\scitools-2db3cbb5076a
```

Installation is done by

Terminal

```
Terminal> python setup.py install
```

4 VMWare Fusion virtual machine

A virtual machine allows you to run another complete computer system in a separate window. For Mac users, I recommend VMWare Fusion over VirtualBox for running a Linux (or Windows) virtual machine. (VMWare Fusion's hardware integration seems superior to that of VirtualBox.) VMWare Fusion is commercial software, but there is a free trial version you can start with. Alternatively, you can use the simpler VMWare Player, which is free for personal use.

4.1 Installing Ubuntu

The following recipe will install a Ubuntu virtual machine under VMWare Fusion.

1. Download Ubuntu²⁰. Choose a version that is compatible with your computer, usually a 64-bit version nowadays.
2. Launch VMWare Fusion (the instructions here are for version 7).
3. Click on *File - New* and choose to *Install from disc or image*.
4. Click on *Use another disc or disc image* and choose your *.iso* file with the Ubuntu image.
5. Choose *Easy Install*, fill in password, and check the box for sharing files with the host operating system.
6. Choose *Customize Settings* and make the following settings (these settings can be changed later, if desired):
 - *Processors and Memory*: Set a minimum of 2 Gb memory, but not more than half of your computer's total memory. The virtual machine can use all processors.
 - *Hard Disk*: Choose how much disk space you want to use inside the virtual machine (20 Gb is considered a minimum).
7. Choose where you want to store virtual machine files on the hard disk. The default location is usually fine. The directory with the virtual machine files needs to be frequently backed up so make sure you know where it is.
8. Ubuntu will now install itself without further dialog, but it will take some time.
9. You may need to define a higher resolution of the display in the Ubuntu machine. Find the *System settings* icon on the left, go to *Display*, choose some display (you can try several, click *Keep this configuration* when you are satisfied).

²⁰<http://www.ubuntu.com/desktop/get-ubuntu/download>

10. You can have multiple keyboards on Ubuntu. Launch *System settings*, go to *Keyboard*, click the *Text entry* hyperlink, add keyboard(s) (*Input sources to use*), and choose a shortcut, say **Ctrl+space** or **Ctrl+backslash**, in the *Switch to next source using* field. Then you can use the shortcut to quickly switch keyboard.
11. A terminal window is key for programmers. Click on the Ubuntu icon on the top of the left pane, search for **gnome-terminal**, right-click its new icon in the left pane and choose **Lock to Launcher** such that you always have the terminal easily accessible when you log in. The **gnome-terminal** can have multiple tabs (**Ctrl+shift+t** to make a new tab).

4.2 Installing software on Ubuntu

You now have a full Ubuntu machine, but there is not much software on it for doing scientific computing with Python. Installation is performed through the Ubuntu Software Center (a graphical application) or through Unix commands, typically

Terminal

```
Terminal> sudo apt-get install packagename
```

To look up the right package name, run **apt-cache search** followed by typical words of that package. The strength of the **apt-get** way of installing software is that the package *and all packages it depends on* are automatically installed through the **apt-get install** command. This is in a nutshell why Ubuntu (or Debian-based Linux systems) are so user-friendly for installing sophisticated mathematical software.

To install a lot of useful packages for scientific work, go to <http://goo.gl/RVHixr> and click on *one* of the following files, which will install a collection of software for scientific work using **apt-get**:

- **install_minimal.sh**: install a minimal collection
- **install_rich.sh**: install a rich collection (takes time to run)

Then click the *Raw* button. The file comes up in the browser window, right-click and choose *Save As...* to save the file on your computer. The next step is to find the file and run it:

Terminal

```
Terminal> cd ~/Downloads
Terminal> bash install_minimal.sh
```

The program will run for quite some time, hopefully without problems. If it stops, set a comment sign **#** in front of the line where it stopped and rerun.

4.3 File sharing

The Ubuntu machine can see the files on your host system if you download *VMWare Tools*. Go to the *Virtual Machine* pull-down menu in VMWare Fusion and choose *Install VMWare Tools*. A tarfile is downloaded. Click on it and it will open a folder `vmware-tools-distrib`, normally in your home folder. Move to the new folder and run `sudo perl vmware-install.pl`. You can go with the default answers to all the questions.

On a Mac, you must open *Virtual Machine - Settings...* and choose *Sharing* to bring up a dialog where you can add the folders you want to be visible in Ubuntu. Just choose your home folder. Then turn on the file sharing button (or turn off and on again). Go to Ubuntu and check if you can see all your host system's files in `/mnt/hgfs/`.

If you later detect that `/mnt/hgfs/` folder has become empty, VMWare Tools must be reinstalled by first turning shared folders off, and then running

Terminal

```
Terminal> sudo /usr/bin/vmware-config-tools.pl
```

Occasionally it is necessary to do a full reinstall by `sudo perl vmware-install.pl` as above.

Backup of a VMWare virtual machine on a Mac.

The entire Ubuntu machine is a folder on the host computer, typically with a name like `Documents/Virtual Machines/Ubuntu 64-bit`. Backing up the Ubuntu machine means backing up this folder. However, if you use tools like Time Machine and work in Ubuntu during backup, the copy of the state of the Ubuntu machine is likely to be corrupt. You are therefore strongly recommended to shut down the virtual machine prior to running Time Machine or simply copying the folder with the virtual machine to some backup disk.

If something happens to your virtual machine, it is usually a straightforward task to make a new machine and import data and software automatically from the previous machine.

5 Dual boot on Windows

Instead of running Ubuntu in a virtual machine, Windows users also have the option of deciding on the operating system when turning on the machine (so-called dual boot). The Wubi²¹ tool makes it very easy to get Ubuntu on a Windows machine this way. There are problems with Wubi on Windows 8, see

²¹<http://wubi-installer.org>

instructions²² for how to get around them. It is also relatively straightforward to perform a direct install of Ubuntu by downloading an Ubuntu image, creating a bootable USB stick on Windows²³ or Mac²⁴, restarting the machine and finally installing Ubuntu²⁵. However, with the powerful computers we now have, a virtual machine is more flexible since you can switch between Windows and Ubuntu as easily as going from one window to another.

6 Vagrant virtual machine

A vagrant machine is different from a standard virtual machine in that it is run in a terminal window on a Mac or Windows computer. You will write programs in Mac/Windows, but run them inside a Vagrant Ubuntu machine that can work with your files and folders on Mac/Windows. This is a bit simpler technology than a full VMWare Fusion virtual machine, as described above, and allows you to work in your original operating system. There is need to install VirtualBox and Vagrant, and on Windows also Cygwin. Then you can download a Vagrant machine with Ubuntu and either fill it with software as explained above, or you can download a ready-made machine. A special machine²⁶ has been made for this book. We also have a larger and richer machine²⁷. The username and password are *fenics*.

Pre-i3/5/7 Intel processors and 32 vs. 64 bit.

If your computer has a pre-i3/5/7 Intel processor and the processor does not have VT-x enabled, you cannot use the pre-packaged 64-bit virtual machines referred to above. Instead, you have to download a plain 32-bit Ubuntu image and install the necessary software (see Section ??). To check if your computer has VT-x (hardware virtualization) enabled, you can use this tool: <https://www.grc.com/securable.htm>.

7 How to write and run a Python program

You have basically three choices to develop and test a Python program:

1. use the IPython notebook
2. use an Integrated Development Environment (IDE), like Spyder, which offers a window with a text editor and functionality to run programs and observe the output

²²<https://www.youtube.com/watch?v=gZqsXAoLBDI>

²³<http://www.ubuntu.com/download/desktop/create-a-usb-stick-on-windows>

²⁴<http://www.ubuntu.com/download/desktop/create-a-usb-stick-on-mac-osx>

²⁵<http://www.ubuntu.com/download/desktop/install-ubuntu-desktop>

²⁶<http://goo.gl/hrdhGt>

²⁷<http://goo.gl/uu5Kts>

3. use a text editor and a terminal window

The IPython notebook is briefly described in Section 9, while the other two options are outlined below.

7.1 The need for a text editor

Since programs consist of plain text, we need to write this text with the help of another program that can store the text in a file. You have most likely extensive experience with writing text on a computer, but for writing your own programs you need special programs, called *editors*, which preserve exactly the characters you type. The widespread word processors, Microsoft Word being a primary example, are aimed at producing nice-looking reports. These programs *format* the text and are *not* acceptable tools for writing your own programs, even though they can save the document in a pure text format. Spaces are often important in Python programs, and *editors* for plain text give you complete control of the spaces and all other characters in the program file.

7.2 Spyder

Spyder is graphical application for developing and running Python programs, available on all major platforms. Spyder comes with Anaconda and some other pre-built environments for scientific computing with Python. On Ubuntu it is conveniently installed by `sudo apt-get install spyder`.

The left part of the Spyder window contains a plain text editor. Click in this window and write `print 'Hello!'` and return. Choose *Run* from the *Run* pull-down menu, and observe the output `Hello!` in the lower right window where the output from programs is visible.

You may continue with more advanced statements involving graphics:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0, 4, 101)
y = np.exp(-x)*np.sin(np.pi*x)
plt.plot(x,y)
plt.title('First test of Spyder')
plt.savefig('tmp.png')
plt.show()
```

Choosing *Run - Run* now leads to a separate window with a plot of the function $e^{-x} \sin(\pi x)$. Figure 1 shows how the Spyder application may look like.

The plot file we generate in the above program, `tmp.png`, is by default found in the Spyder folder listed in the default text in the top of the program. You can choose *Run - Configure ...* to change this folder as desired. The program you write is written to a file `.temp.py` in the same default folder, but any name and folder can be specified in the standard *File - Save as...* menu.

A convenient feature of Spyder is that the upper right window continuously displays documentation of the statements you write in the editor to the left.

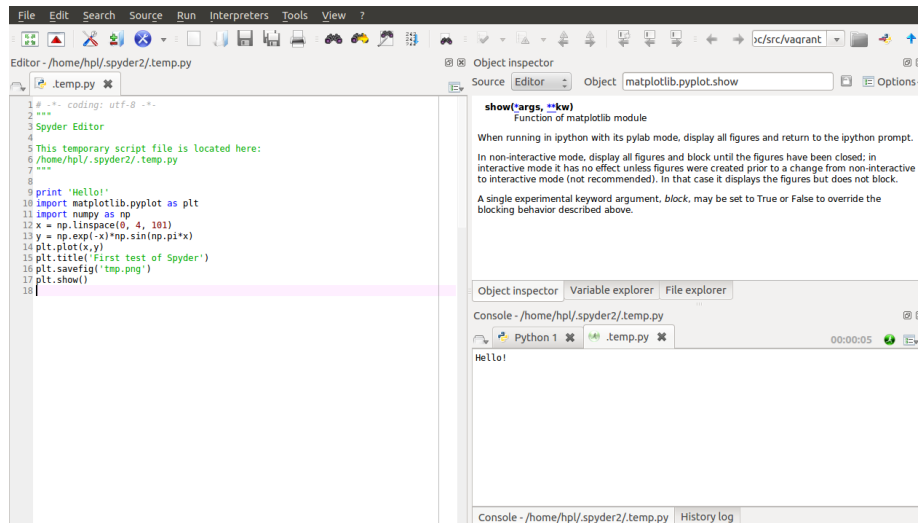


Figure 1: The Spyder Integrated Development Environment.

7.3 Text editors

The most widely used editors for writing programs are Emacs and Vim, which are available on all major platforms. Some simpler alternatives for beginners are

- Linux: Gedit
- Mac OS X: TextWrangler
- Windows: Notepad++

We may mention that Python comes with an editor called Idle, which can be used to write programs on all three platforms, but running the program with command-line arguments is a bit complicated for beginners in Idle so Idle is not my favorite recommendation.

Gedit is a standard program on Linux platforms, but all other editors must be installed in your system. This is easy: just google for the name, download the file, and follow the standard procedure for installation. All of the mentioned editors come with a graphical user interface that is intuitive to use, but the major popularity of Emacs and Vim is due to their rich set of short-keys so that you can avoid using the mouse and consequently edit at higher speed.

7.4 Terminal windows

To run the Python program, you need a *terminal window*. This is a window where you can issue Unix commands in Linux and Mac OS X systems and DOS commands in Windows. On a Linux computer, **gnome-terminal** is my favorite,

but other choices work equally well, such as `xterm` and `konsole`. On a Mac computer, launch the application *Utilities - Terminal*. On Windows, launch *PowerShell*.

You must first move to the right folder using the `cd foldername` command. Then running a python program `prog.py` is a matter of writing `python prog.py`. Whatever the program prints can be seen in the terminal window.

7.5 Using a plain text editor and a terminal window

1. Create a folder where your Python programs can be located, say with name `mytest` under your home folder. This is most conveniently done in the terminal window since you need to use this window anyway to run the program. The command for creating a new folder is `mkdir mytest`.
2. Move to the new folder: `cd mytest`.
3. Start the editor of your choice.
4. Write a program in the editor, e.g., just the line `print 'Hello!'`. Save the program under the name `myprog1.py` in the `mytest` folder.
5. Move to the terminal window and write `python myprog1.py`. You should see the word `Hello!` being printed in the window.

8 The SageMathCloud and Wakari web services

You can avoid installing Python on your machine completely by using a web service that allows you to write and run Python programs. Computational science projects will normally require some kind of visualization and associated graphics packages, which is not possible unless the service offers IPython notebooks. There are two excellent web services with notebooks: *SageMathCloud* at <https://cloud.sagemath.com/> and *Wakari* at <https://www.wakari.io/wakari>. At both sites you must create an account before you can write notebooks in the web browser and download them to your own computer.

8.1 Basic intro to SageMathCloud

Sign in, click on *New Project*, give a title to your project and decide whether it should be private or public, click on the project when it appears in the browser, and click on *Create or Import a File, Worksheet, Terminal or Directory...* If your Python program needs graphics, you need to choose *IPython Notebook*, otherwise you can choose *File*. Write the name of the file above the row of buttons. Assuming we do not need any graphics, we create a plain Python file, say with name `py1.py`. By clicking *File* you are brought to a browser window with a text editor where you can write Python code. Write some code and click *Save*. To run the program, click on the plus icon (*New*), choose *Terminal*, and

you have a plain Unix terminal window where you can write `python py1.py` to run the program. Tabs over the terminal (or editor) window make it easy to jump between the editor and the terminal. To download the file, click on *Files*, point on the relevant line with the file, and a download icon appears to the very right. The IPython notebook option works much in the same way, see Section 9.

8.2 Basic intro to Wakari

After having logged in at the `wakari.io` site, you automatically enter an IPython notebook with a short introduction to how the notebook can be used. Click on the *New Notebook* button to start a new notebook. Wakari enables creating and editing plain Python files too: click on the *Add file* icon in pane to the left, fill in the program name, and you enter an editor where you can write a program. Pressing *Execute* launches an IPython session in a terminal window, where you can run the program by `run prog.py` if `prog.py` is the name of the program. To download the file, select `test2.py` in the left pane and click on the *Download file* icon.

There is a pull-down menu where you can choose what type of terminal window you want: a plain Unix shell, an IPython shell, or an IPython shell with Matplotlib for plotting. Using the latter, you can run plain Python programs or commands with graphics. Just choose the type of terminal and click on *+Tab* to make a new terminal window of the chosen type.

8.3 Installing your own Python packages

Both SageMathCloud and Wakari let you install your own Python packages. To install any package `packagename` available at PyPi²⁸, run

Terminal

```
pip install --user packagename
```

To install the SciTools package, which is useful when working with this book, create a Terminal (with a Unix shell) and run the command

Terminal

```
pip install --user -e \
git+https://github.com/hplgit/scitools.git#egg=scitools
```

9 Writing IPython notebooks

The IPython notebook is a splendid interactive tool for doing science, but it can also be used as a platform for developing Python code. You can

²⁸<https://pypi.python.org/pypi>

either run it locally on your computer or in a web service like SageMath-Cloud or Wakari. Installation on your computer is trivial on Ubuntu, just `sudo apt-get install ipython-notebook`, and also on Windows and Mac²⁹ by using Anaconda or Enthought Canopy for the Python installation.

The interface to the notebook is a web browser: you write all the code and see all the results in the browser window. There are excellent YouTube videos on how to use the IPython notebook, so here we provide a very quick “step zero” to get anyone started.

9.1 A simple program in the notebook

Start the IPython notebook locally by the command `ipython notebook` or go to SageMathCloud or Wakari as described above. The default input area is a *cell* for Python code. Type

```
g = 9.81
v0 = 5
t = 0.6
y = v0*t - 0.5*g*t**2
```

in a cell and *run the cell* by clicking on *Run Selected* (notebook running locally on your machine) or on the “play” button (notebook running in the cloud). This action will execute the Python code and initialize the variables `g`, `v0`, `t`, and `y`. You can then write `print y` in a new cell, execute that cell, and see the output of this statement in the browser. It is easy to go back to a cell, edit the code, and re-execute it.

To download the notebook to your computer, choose the *File - Download as* menu and select the type of file to be downloaded: the original notebook format (`.ipynb` file extension) or a plain Python program version of the notebook (`.py` file extension).

9.2 Mixing text, mathematics, code, and graphics

The real strength of IPython notebooks arises when you want to write a report to document how a problem can be explored and solved. As a teaser, open a new notebook, click in the first cell, and choose *Markdown* as format (notebook running locally) or switch from *Code* to *Markdown* in the pull-down menu (notebook in the cloud). The cell is now a text field where you can write text with Markdown³⁰ syntax. Mathematics can be entered as L^AT_EX code. Try some text with inline mathematics and an equation on a separate line:

Plot the curve $y=f(x)$, where

```
$$
f(x) = e^{-x}\sin(2\pi x), \text{quad } x \text{ in } [0, 4]
$$
```

²⁹<http://ipython.org/install.html>

³⁰<http://daringfireball.net/projects/markdown/syntax>

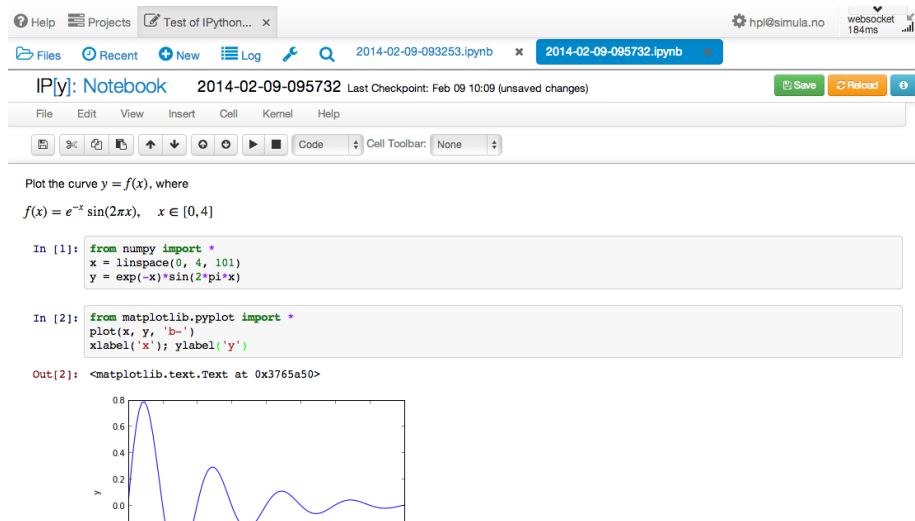


Figure 2: Example on an IPython notebook.

Execute the cell and you will see nicely typeset mathematics in the browser. In the new cell, add some code to plot $f(x)$:

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline # make plots inline in the notebook

x = np.linspace(0, 4, 101)
y = np.exp(-x)*np.sin(2*pi*x)
plt.plot(x, y, 'b-')
plt.xlabel('x'); plt.ylabel('y')
```

Executing these statements results in a plot in the browser, see Figure 2. It was popular to start the notebook by `ipython notebook --pylab` to import everything from `numpy` and `matplotlib.pyplot` and make all plots inline, but the `--pylab` option is now officially discouraged³¹. If you want the notebook to behave more as MATLAB and not use the `np` and `plt` prefix, you can instead of the first three lines above write `%pylab`.

References

- [1] O. Certik et al. SymPy: Python library for symbolic mathematics. <http://sympy.org/>.
- [2] ScientificPython software package. <http://starship.python.net/crew/hinsen>.

³¹<http://carreau.github.io/posts/10-No-PyLab-Thanks.ipynb.html>

- [3] J. D. Hunter. Matplotlib: a 2d graphics environment. *Computing in Science & Engineering*, 9, 2007.
- [4] J. D. Hunter et al. Matplotlib: Software package for 2d graphics. <http://matplotlib.org/>.
- [5] E. Jones, T. E. Oliphant, P. Peterson, et al. SciPy scientific computing library for Python. <http://scipy.org>.
- [6] H. P. Langtangen and J. H. Ring. SciTools: Software tools for scientific computing. <http://code.google.com/p/scitools>.
- [7] T. E. Oliphant. Python for scientific computing. *Computing in Science & Engineering*, 9, 2007.
- [8] T. E. Oliphant et al. NumPy array processing package for Python. <http://www.numpy.org>.
- [9] F. Perez and B. E. Granger. IPython: a system for interactive scientific computing. *Computing in Science & Engineering*, 9, 2007.
- [10] F. Perez, B. E. Granger, et al. IPython software package for interactive scientific computing. <http://ipython.org/>.
- [11] Python programming language. <http://python.org>.

Index

Emacs, 9

Gedit, 9

Idle, 9

Notepad++, 9

TextWrangler, 9

Vim, 9