

PY410 / 505
Computational Physics 1

Salvatore Rappoccio

C++ and python

- We've now looked at how to do C++ and python separately
 - (Technically we also looked at them together, since numpy is written in C++, but more on that later)
- How can we put them together?
- Lots of options out there
 - We'll use SWIG for a concrete example
 - <http://www.swig.org>
 - Tutorial: <http://www.swig.org/tutorial.html>
- We're also going to use Jupyter for this for an additional bit of fun!

C++ and python... with JUPYTER!

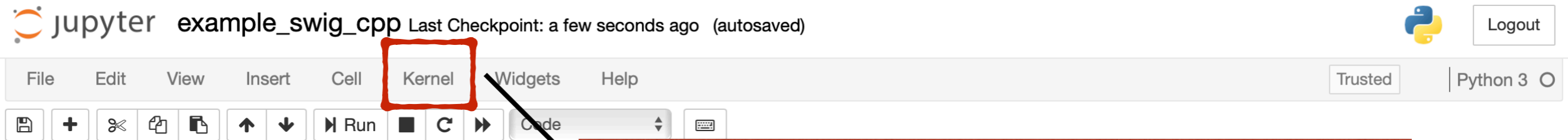
- Then click through to “CompPhys/SwigExamples” and you get something like this:

The screenshot shows the Jupyter web interface. At the top left is the Jupyter logo. On the top right are 'Quit' and 'Logout' buttons. Below the logo are tabs for 'Files', 'Running', and 'IPython Clusters'. A blue box with the text 'Execute this one' is overlaid on the 'Running' tab. Below the tabs is a message: 'Select items to perform actions on them.' To the right of this message are 'Upload', 'New', and a refresh icon. Below this is a table of files and folders. The table has columns for 'Name', 'Last Modified', and 'File size'. The first row is a folder named '..' with a last modified time of 'seconds ago'. The second row is a folder named 'swig_example' with a last modified time of '20 minutes ago'. The third row is a file named 'example_swig_cpp.ipynb' with a status of 'Running', a last modified time of '4 minutes ago', and a file size of '9.11 kB'. An arrow points from the 'Execute this one' box to the 'example_swig_cpp.ipynb' file.

	Name ↓	Last Modified	File size
<input type="checkbox"/>	..	seconds ago	
<input type="checkbox"/>	swig_example	20 minutes ago	
<input type="checkbox"/>	example_swig_cpp.ipynb	Running 4 minutes ago	9.11 kB

C++ and python... with JUPYTER!

- Should look something like this:



Kernel/ Restart and Run All

Example of using C++ code in python and Jupyter with SWIG

It is also possible to use our C++ code from python and Jupyter. This involves using the [SWIG](#) package. You can download it [here](#) and then install following instructions [here](#). If you are successful, you should be able to open a new terminal and type `which swig` to obtain the path of swig.

The idea is then to use SWIG to automatically generate python-readable code from our C++/C libraries. There is a lot to learn in this regard, so we will try first with a simple example that illustrates some concepts we will need, such as using STL libraries and C++11 compilation.

Step 1 : Look at C++ files

You should be able to see these two simple C++ files:

```
In [ ]: ! cat swig_example/example.hpp swig_example/example.cpp
```

Step 2 : Look at SWIG interface file

The magic of SWIG is to create wrapper C++ functions that use the "cython" interface. You will see an "interface" file for SWIG :

```
In [ ]: ! cat swig_example/example.i
```