classy - The Python wrapper

Thomas Tram

Institute of Gravitation and Cosmology

October 27, 2014

Compiled and interpreted languages

Compiled languages

- The code must be **compiled** before it can be executed.
- It will often be **faster**...
- but **less flexible** since some decisions can not be made at runtime.
- C, C++, Fortran, ...

Interpreted languages

- The code is **interpreted** during execution.
- It can be **slower** but is **very flexible**.
- MATLAB, Octave, IDL, Python, ...

Interfacing C with Python

Cython

- **Cython** is a **compiled** language.
- It understands most **Python** syntax.
- It can directly call **external** C libraries, such as `libclass.a`.
- It produces a **Python** module.

classy, the CLASS wrapper

- Written in **Cython**.
- Automatically compiled and installed when you type `make`.
- Needed for MONTE PYTHON and when using CLASS from **Python**.

The CLASSY module

classy, the CLASS wrapper

- All the functionality of **classy** is found in the **Python class called Class**.
- In **Python** import **Class** by: `from classy import Class`

Running CLASS from Python

```
from classy import Class
import numpy as np
import matplotlib.pyplot as plt
```

```
cosmo = Class()
cosmo.set({'output':'tCl,pCl,lCl','lensing':'yes','
    modes':'s,t','r':'0.2'})
cosmo.compute()
cosmo.cleanup()
```

Exercise 1b in Python 1/2

Solving exercise 1b in Python

```python
from classy import Class
import numpy as np
import matplotlib.pyplot as plt
```

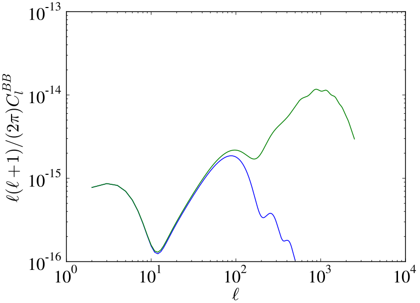
```python
cosmo = Class()
cosmo.set({'output':'tCl,pCl,lCl','lensing':'yes','
    modes':'s,t','r':'0.2'})
cosmo.compute()
cosmo.cleanup()
```

```python
l = np.array(range(2,2501))
factor = l*(l+1)/(2*np.pi)
raw_cl = cosmo.raw_cl(2500)
lensed_cl = cosmo.lensed_cl(2500)
raw_cl.viewkeys()
```

Exercise 1b in Python 2/2

Solving exercise 1b in Python

```
plt.loglog(l,factor*raw_cl[’bb’][2:],l,factor*
    lensed_cl[’bb’][2:])
plt.xlabel(r"$\ell$")
plt.ylabel(r"$\ell(\ell+1)/(2\pi) C_l^{BB}$")
plt.tight_layout()
plt.savefig("solution1b.pdf")
```



Interactive mode

IPython Notebook

IPython Notebook is a Mathematica style (cell) interface to IPython.

- Has Tab-completion of variables and function names.
- Nicely presents the documentation of each function.
- Easy way to get started on Python.

Interactive mode

Launching IPython Notebook

Write the following command to launch the notebook:

```
ipython notebook --pylab=inline
  --InlineBackend.figure_format=svg
```

You probably want to alias this command to e.g. `inote`.

You can open an existing notebook by

```
inote MyFirstCLASSNotebook.ipynb
```

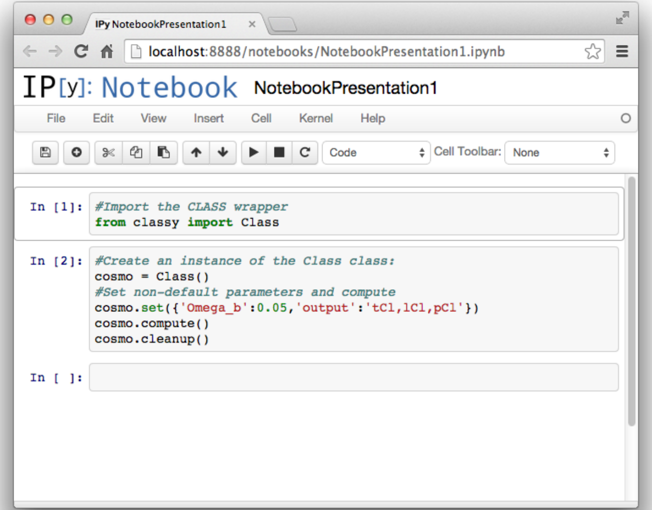
Aliasing

To make an alias, open your shell {bash, zsh, ...} startup script: {~\.bashrc, ~\.zshrc, ...}

At the bottom of the file, add the line

```
alias inote="some command"
```

The notebook



In [1]:

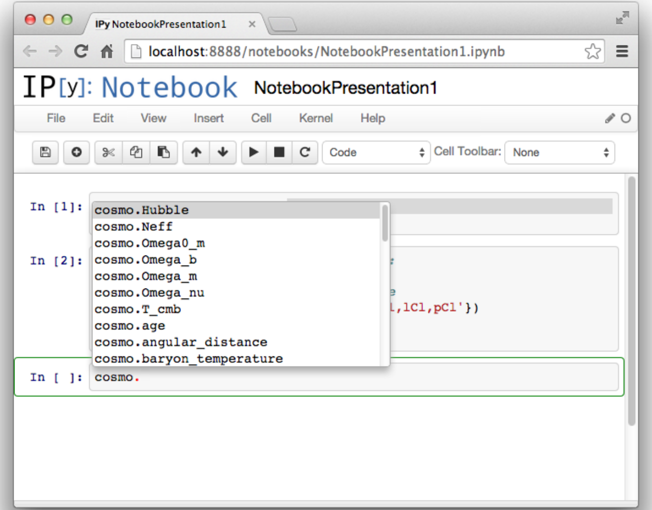
```
#Import the CLASS wrapper
from classy import Class
```

In [2]:

```
#Create an instance of the Class class:
cosmo = Class()
#Set non-default parameters and compute
cosmo.set({'Omega_b':0.05,'output':'tCl,lCl,pCl'})
cosmo.compute()
cosmo.cleanup()
```

In [ ]:

Tab: Available class methods

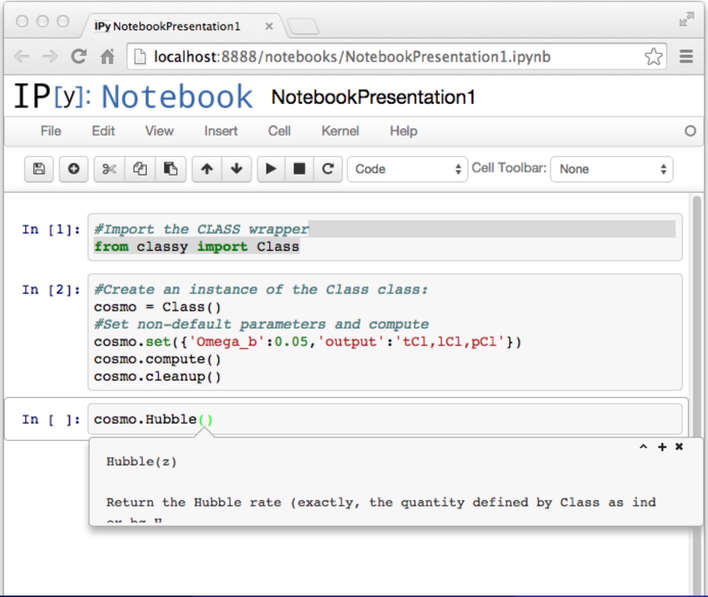


```
In [1]:

In [2]:
                                                 l,lCl,pCl'})
```

```
cosmo.Hubble
cosmo.Neff
cosmo.Omega0_m
cosmo.Omega_b
cosmo.Omega_m
cosmo.Omega_nu
cosmo.T_cmb
cosmo.age
cosmo.angular_distance
cosmo.baryon_temperature
```

```
In [ ]: cosmo.
```

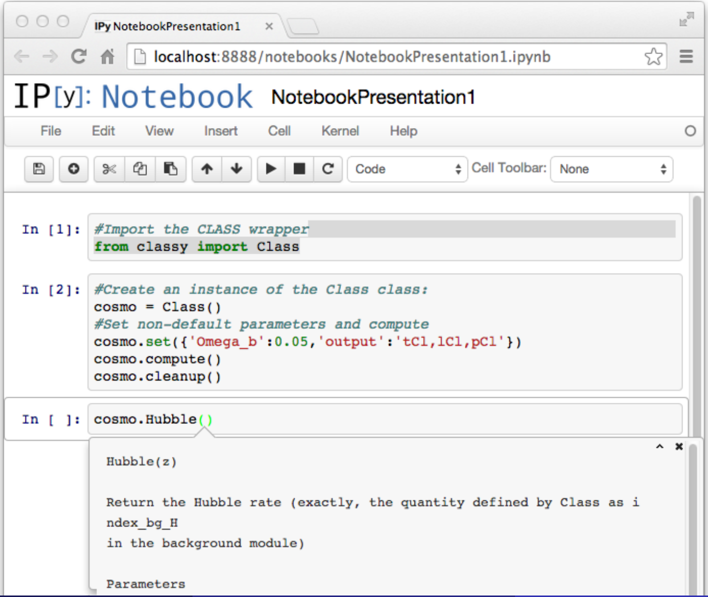
Shift+Tab: Help on method



```
In [1]: #Import the CLASS wrapper
        from classy import Class

In [2]: #Create an instance of the Class class:
        cosmo = Class()
        #Set non-default parameters and compute
        cosmo.set({'Omega_b':0.05,'output':'tCl,lCl,pCl'})
        cosmo.compute()
        cosmo.cleanup()

In [ ]: cosmo.Hubble()
```

```
Hubble(z)

Return the Hubble rate (exactly, the quantity defined by Class as ind
```

Shift+Tab: More help



```
In [1]: #Import the CLASS wrapper
        from classy import Class

In [2]: #Create an instance of the Class class:
        cosmo = Class()
        #Set non-default parameters and compute
        cosmo.set({'Omega_b':0.05,'output':'tCl,lCl,pCl'})
        cosmo.compute()
        cosmo.cleanup()

In [ ]: cosmo.Hubble()
```

```
Hubble(z)

Return the Hubble rate (exactly, the quantity defined by Class as i
ndex_bg_H
in the background module)

Parameters
```

What can it do?

What is available in the wrapper?

- `get_background()` returns the information normally found in `_background.dat`.
- `get_thermodynamics()` returns the information of `_thermodynamics.dat`.
- `get_primordial()` corresponds to `_primordial_Pk.dat`.
- `get_perturbations()` returns everything found in `_perturbations*.dat`
- `get_transfer(z,format)` returns the density and velocity transfer functions at any* redshift z. (Format can be either `'camb'` or `'class'`).

And even more...

What is available in the wrapper?

- `raw_cl()` returns unlensed C_ℓ.
- `lensed_cl()` returns lensed C_ℓ.
- `density_cl()` returns density C_ℓ.
- `pk(k, z)` returns the $P(k)$ at redshift z.
- Many other small functions.

Exercise!

IPython Notebook exercise

Try to solve some or all of exercise 1a-1d from yesterday using the IPython Notebook.

Example notebooks

Play around with some of the example notebooks found in IPythonNotebooks folder on Dropbox.