Python (for physicists)

What is Python?

- *Interpreted* programming language
- First release in 1991 (~17 years ago!)
- "Emphasizes programmer **productivity** and **code readability**"
- Quick turnaround in code development
- Reduces complexities
- Improves code exchange among developers

Why using Python?

- Because it recommends one and only one developing style it eases code exchange among people
- *(Automated tool for correcting the style by giving a nice mark like in the good old school days...))*
- Is integrated with physics analysis frameworks (e.g. ROOT via PyROOT)
- Very similar to pseudo-language: let you transform almost immediately ideas into working code

```
def is_nice_word(word):
    if word in ('nice', 'beautiful', 'good'):
        print '%s is a nice word' % word
    else:
        print '%s is not a nice word!' % word
```

e.g.: Reading Text

C++

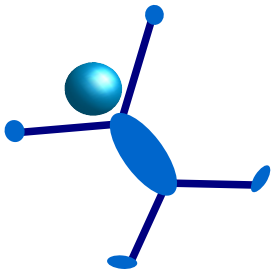
```cpp
// reading a text file
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

void reader (string &filename){
  string line;
  ifstream myfile(filename);
  while (!myfile.eof()) {
    getline (myfile,line);
    cout << line << endl;
  }
}
```

Python

```python
# reading a text file
def reader(filename):
    for line in file(filename):
        print line
```

You have space to do even more!

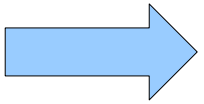
```python
def read_numbers(filename):
    numbers = []
    for line in file(filename):
        line = line.split(',')
        numbers.append(line)
    return numbers
```



e.g.: Configuration files

- They are very common for storing parameters of a long task (and keeping track of them)
- In Python you can use **JSon**

```
{
    “option1” : [2008, 8],
    “option2” : “CSC”
}
```



```
> import simplejson
> simplejson.load(open('test.cfg'))
  {u'option1': [2008, 8],
  u'option2': u'CSC'}
```

e.g.: Logging

- For those used to debug via printing...
- A full and easy to use framework for logging
- To file a/o to screen, hierarchical, compressed, rotation, extensible...

```
from logging import debug, info, warning, error

def my_function(parameters):
    try:
        debug("Parameters: %s" % parameters)
        info("Very long computation started...")
        [...]
        if bad_thing_p:
            warning("Bad thing has happened!")
    except Exception, e:
        error("An unexpected exception has happened: %s" % e)
        raise
    info("Very long computation terminated.")
```

Integration with ROOT

- You can use Python for your analysis script and still *rely on the powerful ROOT framework* for the most computational demanding algorithms.

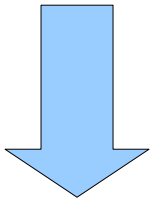
```
from ROOT import TH1F

histo = TH1F(...)
[...]
```

- Everything that you were used to find in ROOT C++ is available in Python too! (Including your C++ custom code)

```
import ROOT

ROOT.gROOT.ProcessLine(".L my_class.C+")
my_object = ROOT.MyClass()
```

Performance

- Being an *interpreted language* Python is of course slower than C/C++
- If you want to have the best of the two worlds (and you're not already using ROOT), there comes **Cython**
- Write your power demanding algorithms in C/C++ and wrap them with a simple code.



- They'll be available as normal Python functions or classes!

Conclusions

- Python is **easy**!
- Trivially glue your scripts to your C++ code and framework today!

References:
http://www.python.org
http://www.cython.org
http://www.json.org