EuroPython 2011

# PyF A dataflow processing framework

Dataflow processing, mining, transforming and reporting using Python.


Presented by :
Jonathan Schemoul


jmsi Software Consulting and Services

PYthon Flow

# What is dataflow programming ?

- In the 70s: John Paul Morrison in Canada
  - Bank applications
- We process data one by one
  - Items pass from one block to another, creating a flow
  - We transform (adapt) items in the flow like in a production chain.

# What is dataflow programming ?

In the theory :

- All the blocks (components) make a network
- Components have input and output
- Network in managed by a scheduler
  - Checks network state, manage processes
- Data sent as information packets

# Dataflow programming with Python

We have generators

Code execution gets resumed at next item

What if we tried to use that for dataflow programming ?

```
>>> def my_generator(size=100) :
...     for i in range(size) :
...         print « I continue », i
...         yield dict(iteration=i,
...                    answer=42)

>>> iterator = my_generator(size=5)

>>> print repr(iterator.next())

I continue 0
{'iteration': 0, 'answer': 42}

>>> for value in iterator :
...     print repr(value)

I continue 1
{'iteration': 1, 'answer': 42}
I continue 2
{'iteration': 2, 'answer': 42}
I continue 3
{'iteration': 3, 'answer': 42}
I continue 4
{'iteration': 4, 'answer': 42}
```

# Chaining generators

To use generators for dataflow programming...

We define the blocks , with generators

- We pass the iterator to another generator, making a pipe
- We « adapt » the flow with each block

```
>>> def my_adapter(flux) :
...     for v in flux:
...         print « I received », v
...         yield dict(value=v,
...                    double=v*2)

>>> it1 = xrange(5) # 0 to 5
>>> it2 = my_adapter(it1)

>>> def my_consumer(flux_adapted) :
...     for nv in flux_adapted:
...         print nv['double']
...         yield True # It's ok.

>>> it3 = my_consumer(it2)

>>> for value in it3:
...     if value is True:
...         print 'ok so far'
...     else:
...         print 'not ok'

I received 0
0
ok so far
I received 1
2
ok so far
[...]
```

# Just a few words

- Blocks (components) in a process make a network
- Each component has input and output ports
- The network is managed using generators (lazy)
- Transferred data can be either standard python objects or « Information Packets ».

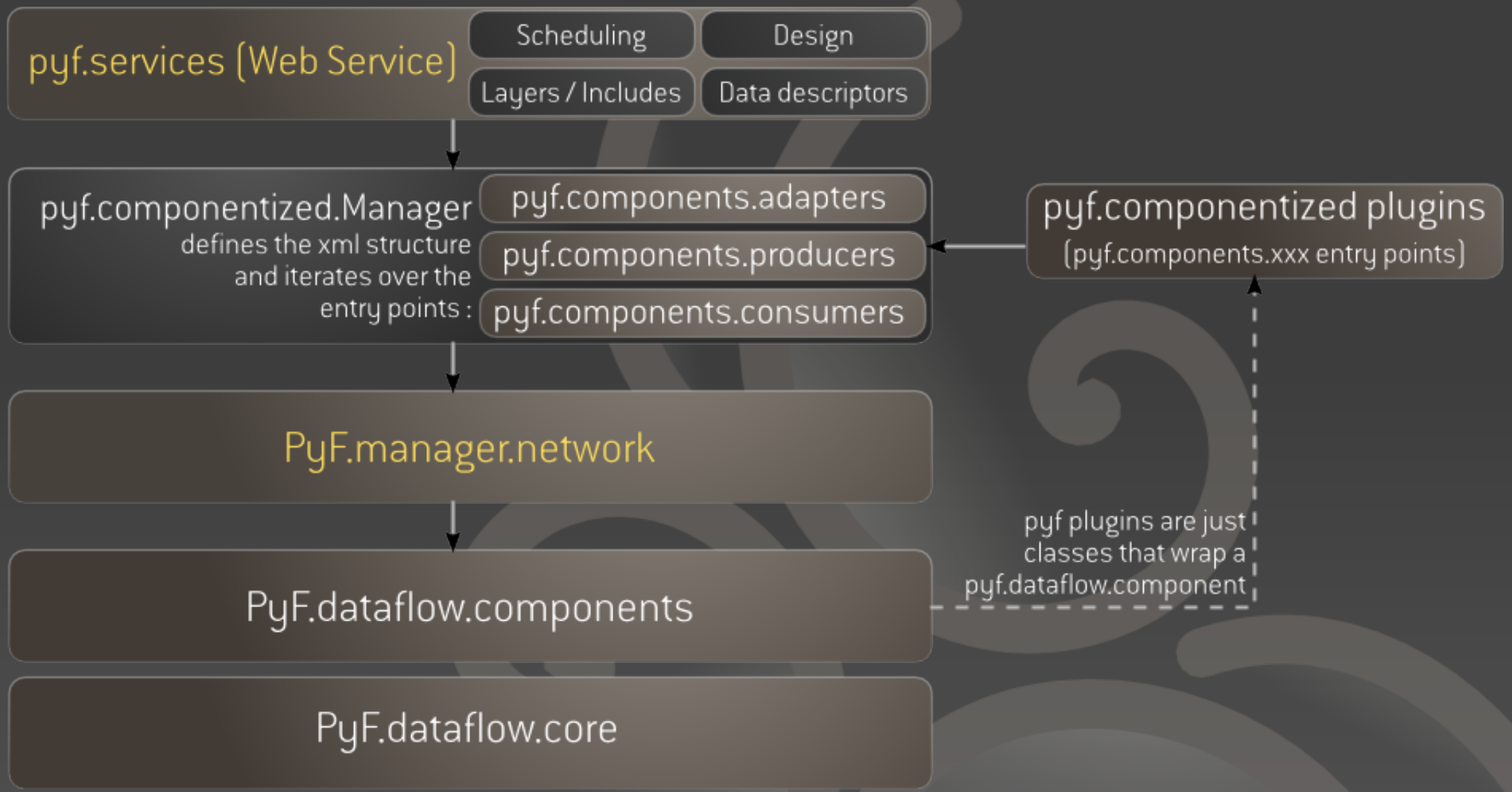
# Benefits using lazy flow programming with PyF

- RAM usage is kept low even with millions of items
- Easy to learn programmation paradigm (you take one item, you give one item or several, or none, you choose! :-))
- All python objects can be used (not only packets)
- No threads
- Just standard plain python (no stackless)
- Need to scale ? Just ask for multiprocessing !

# PyF in a few words

- Development framework and environment for dataflow programming
- Based on Zflow (the low-level core is a « friendly fork »)
- Much more than the basis : « batteries included ».

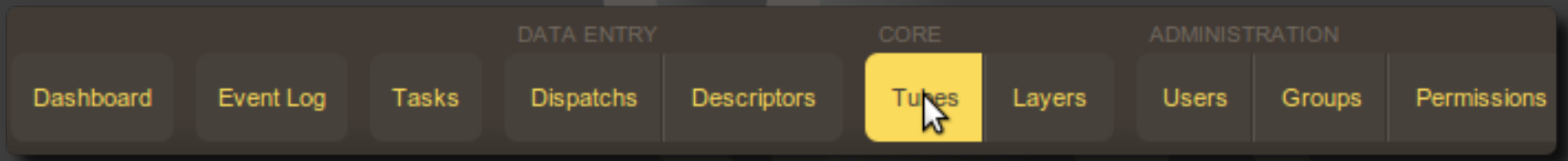
# PyF Architecture



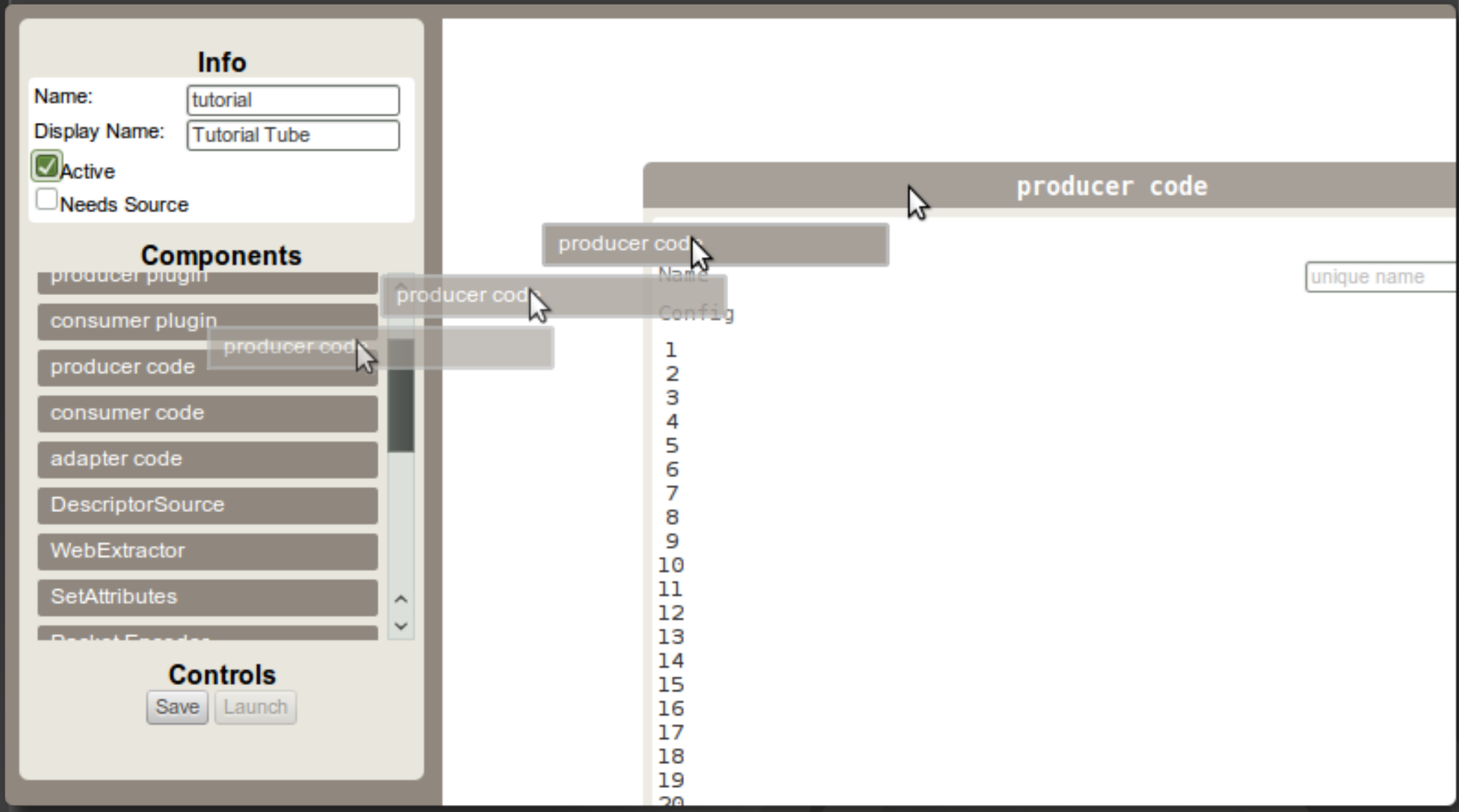
# PyF in detail

- Plugins system
  - A lot of existing plugins
    - Extraction (website, rss, sql alchemy...),
    - Adaptation (summarizing, attribute getters, ...),
    - Writing (csv, xml, fixed-length, xlsx, pdf with xhtml, odt or rml templating, etc.)
- Python (and soon javascript and ruby) code can be inserted directly in tube source
- A lot of other tools (visual designer, scheduling, event logs...)

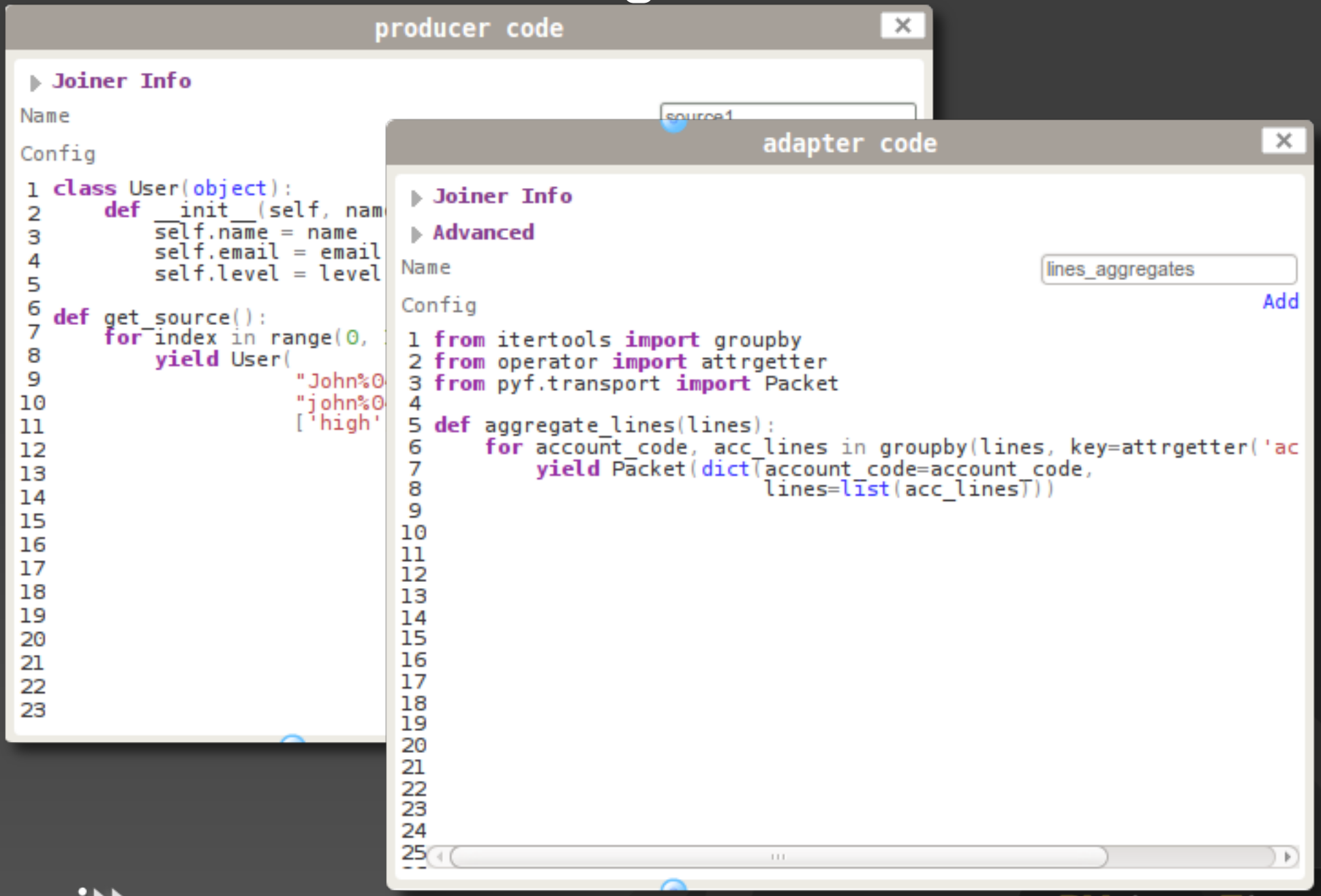
# The Web Service



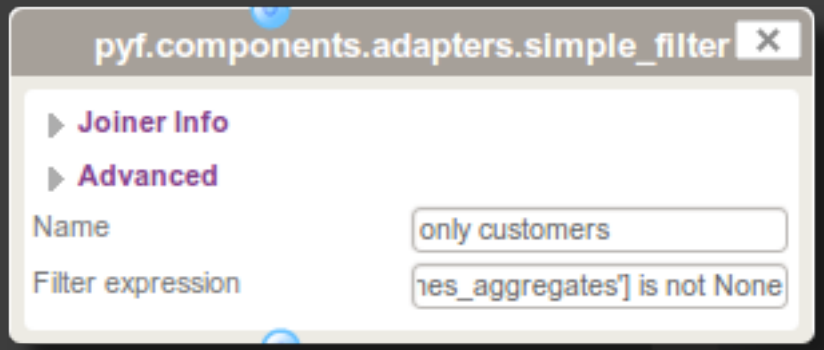
# Creating a tube

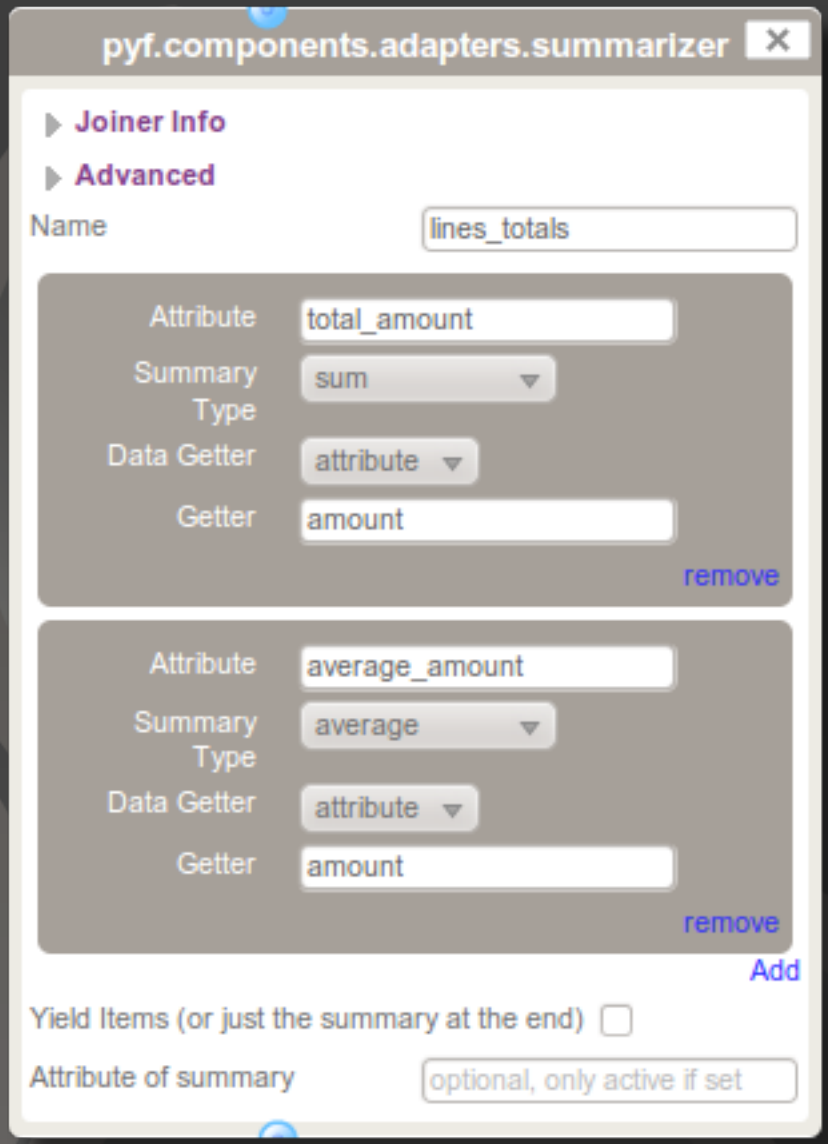


# Using Code

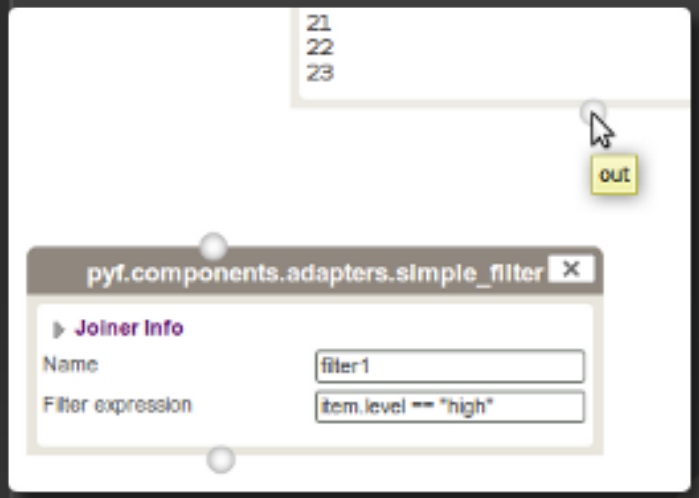

producer code
Joiner Info
Name
source1
Config
1 class User(object):
2 def __init__(self, nam
3 self.name = name
4 self.email = email
5 self.level = level
6
7 def get_source():
8 for index in range(0,
9 yield User(
10 "John%0
11 "john%0
12 ['high'
13
14
15
16
17
18
19
20
21
22
23
adapter code
Joiner Info
Advanced
Name
lines_aggregates
Config
Add
1 from itertools import groupby
2 from operator import attrgetter
3 from pyf.transport import Packet
4
5 def aggregate_lines(lines):
6 for account_code, acc_lines in groupby(lines, key=attrgetter('ac
7 yield Packet(dict(account_code=account_code,
8 lines=list(acc_lines)))
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

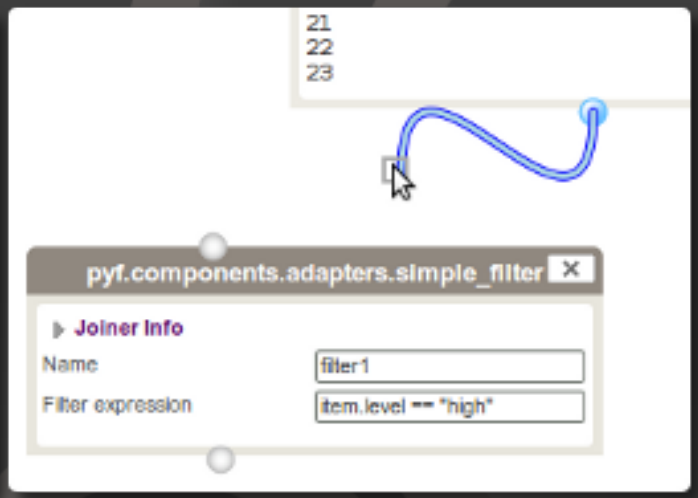
# Or plugins

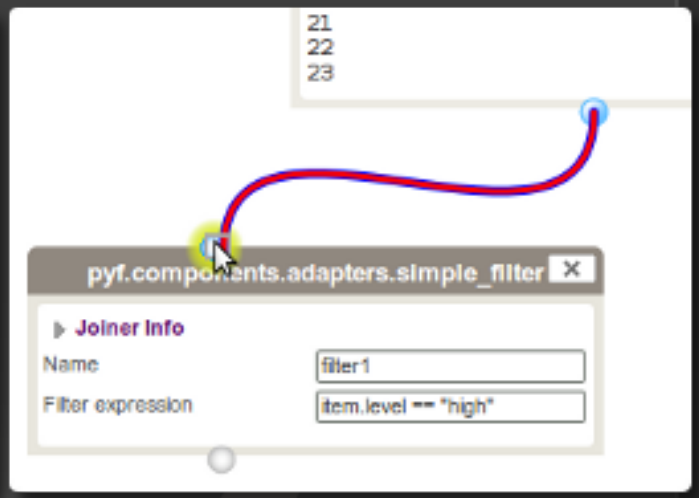

pyf.components.adapters.simple_filter
Joiner Info
Advanced
Name
only customers
Filter expression
nes_aggregates'] is not None

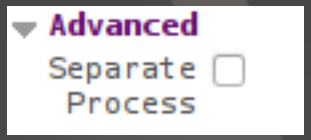

pyf.components.adapters.summarizer
Joiner Info
Advanced
Name
lines_totals
Attribute
total_amount
Summary Type
sum
Data Getter
attribute
Getter
amount
remove
Attribute
average_amount
Summary Type
average
Data Getter
attribute
Getter
amount
remove
Add
Yield Items (or just the summary at the end)
Attribute of summary
optional, only active if set

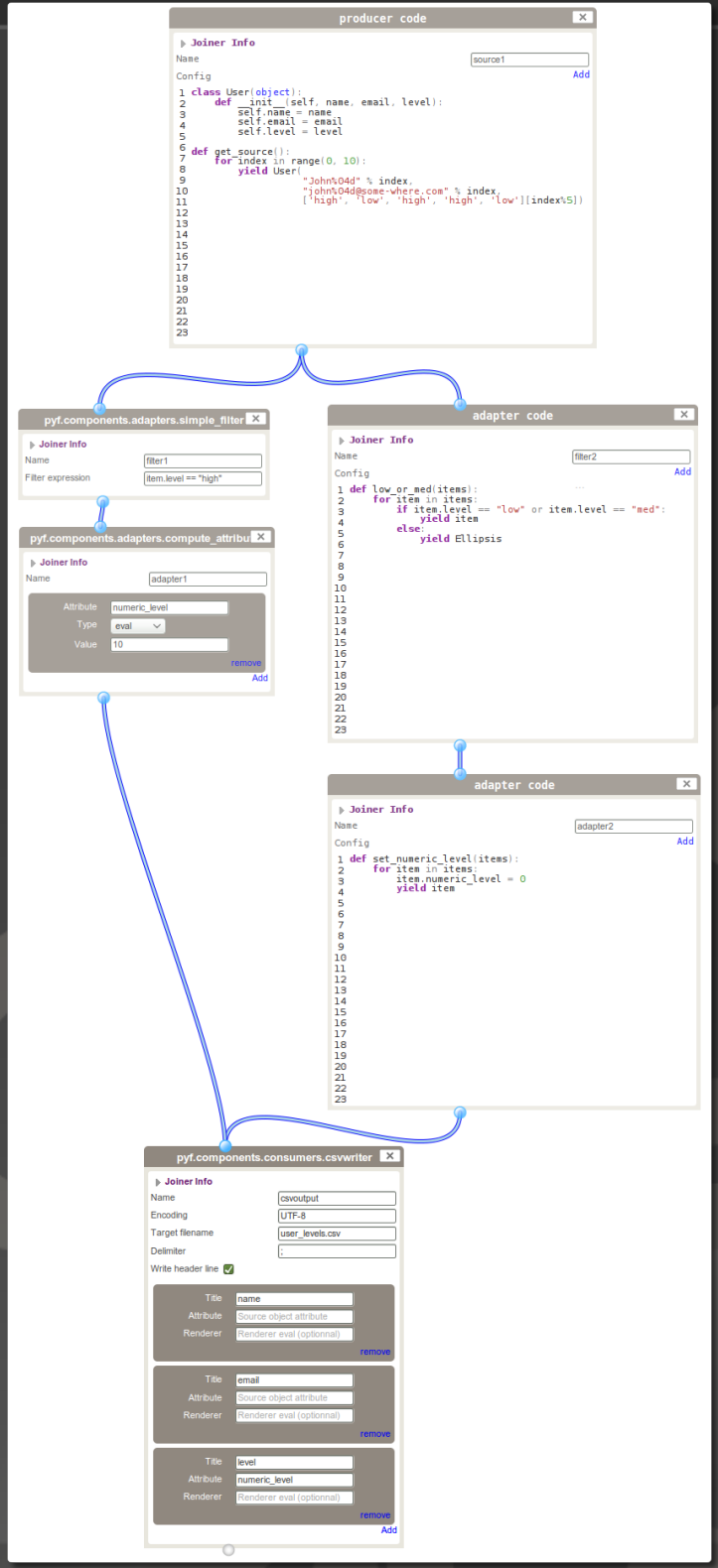
# Link and arrange your nodes...



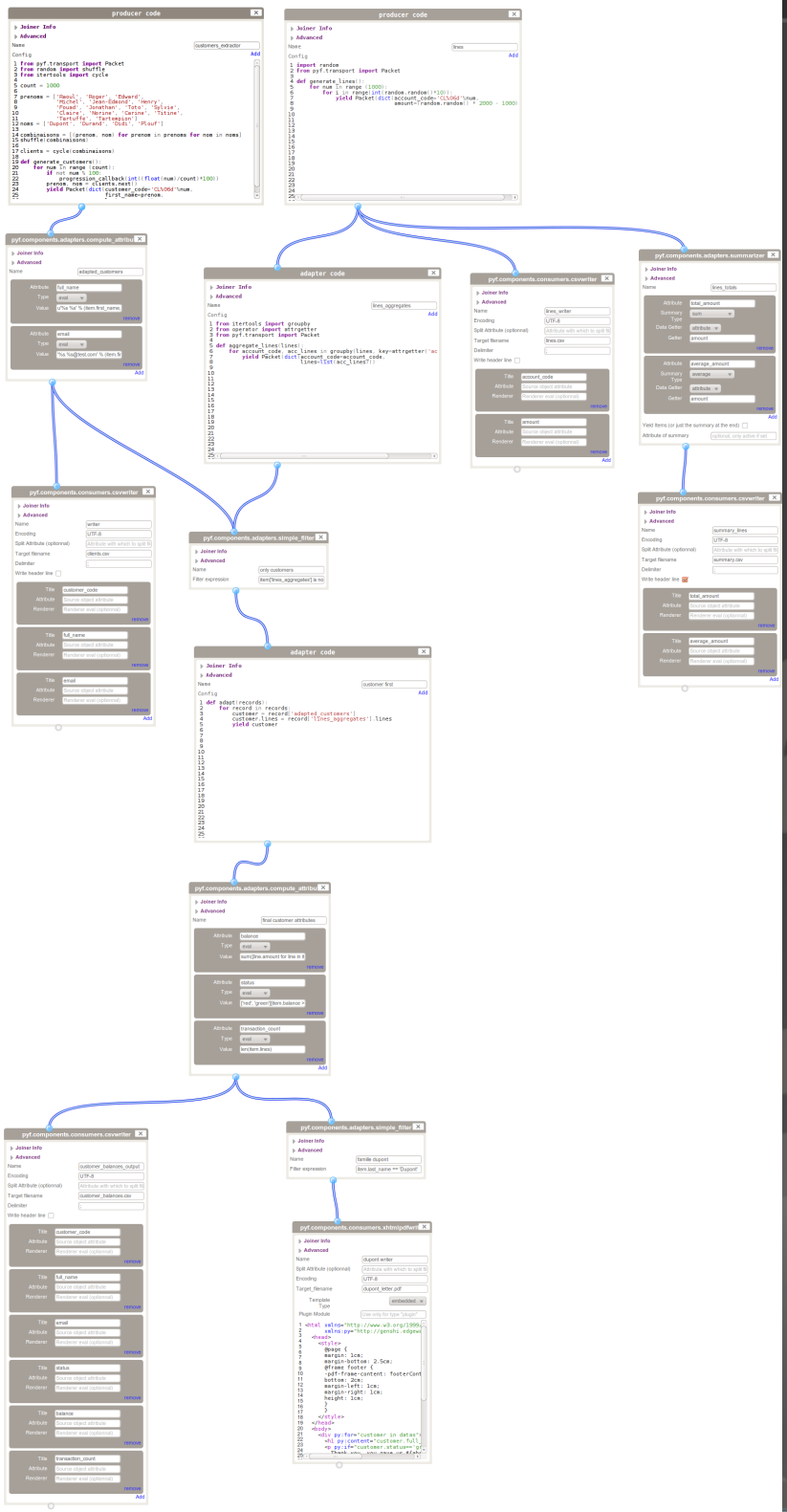






jmsi Software Consulting

PYthon Flow




jmsi Software Consu

PYthon Flow

- Thank you !
- Training is tomorrow at 14:30 in Pizza Napoli
- For more information :

  *http://www.pyfproject.org/*

  *#pyf* on irc.freenode.net

  *http://groups.google.com/group/pyf-users*

- *Business Cards available*

# PyF Sponsors


http://www.xcg-consulting.fr


http://www.jmsinfor.com

# Training Info

- Tomorrow at 14:30 (2:30 PM) at Pizza Napoli
- Please have :
  - Under linux : libxml2-dev libxslt-dev
  - Installed virtualenv (« easy_install -UZ virtualenv »)
- Recommended :
  - Predownload pyf in a new virtualenv:
    - « easy_install -UZ pyf[fullstack] »