Jupyter

IRuby Notebook

RORLAB 84번째 모임

2015. 4. 14.

@nacyo_t

낚읏

가 각 간 갓

나 낙 난 낫

다 닥 단 닷

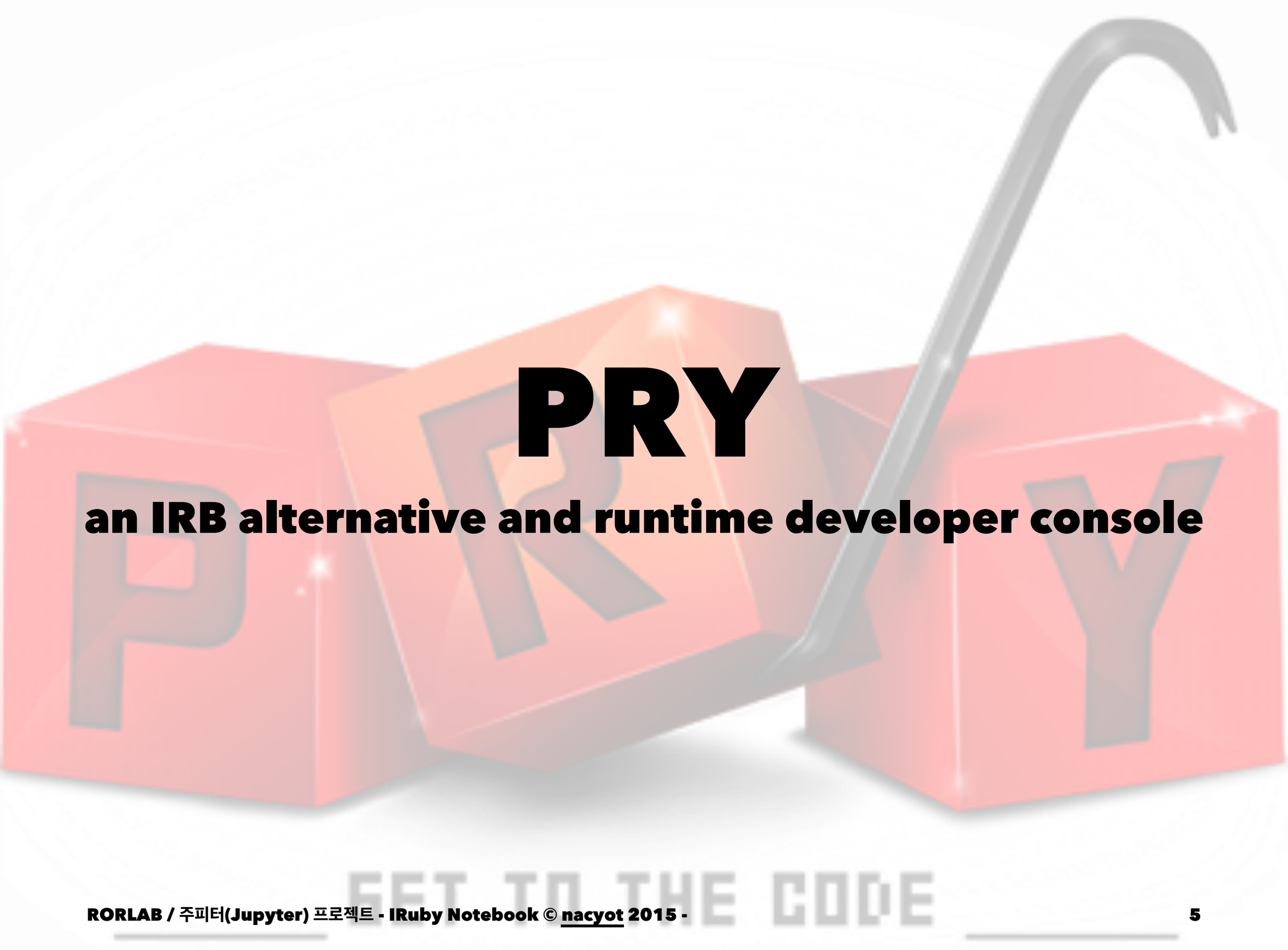
ㄱ ㄴ ㅅ

(ㄱ,ㄴ,ㅅ)x(가,나,다)

REPL of Ruby

irb(루비 내장 REPL)

PRY (확장 REPL)


PRY
an IRB alternative and runtime developer console
GET TO THE CODE

IRuby Notebook

sciruby/iruby

Demo #1

- Executing IRuby Notebook server
- IRuby Notebook interface
- Evaluating simple Ruby expressions

What is REPL?

Read-eval-print loop

from Lisp

IRuby Notebook

2012. 3. 25. ~

IPython

2001년부터 만들어진 Python REPL의 확장

IPython 0.0.1

https://gist.github.com/fperez/1579699

```
- The following GLOBAL variables always exist (so don't overwrite them!):
  _p: stores previous result which generated printable output.
  _pp: next previous
  _ppp: next-next previous
  _cache[]: cache of all previous output, indexed by prompt counter.
  _pc: short alias for _cache
```

Python의 REPL 구현

- 기본 REPL
  - python
- 확장 REPL
  - ipython
  - bpython

왜 Python REPL의 확장을 만들었을까?

왜 Python REPL의 확장을 만들었을까?

I started using Python in 2001 and liked the language, but its interactive prompt felt like a crippled toy compared to the systems mentioned(maple, mathematica, etc) above or to a Unix shell.

– Fernando Perez

루비에서의 pry

```
➜ ir  irb
irb(main):001:0> def hello(name)
irb(main):002:1>   puts "Hello, " + name
irb(main):003:1> end
=> :hello
irb(main):004:0> hello("john")
Hello, john
=> nil
irb(main):005:0> "uppercase".upc^C
irb(main):005:0> "uppercase".upcase
=> "UPPERCASE"
irb(main):006:0> `ls`
=> "comple.rb\nfile\npath.rb\ntest.rb\n"
irb(main):007:0>
```

```
|[1] pry(main)> def hello(name)
|[1] pry(main)*   puts "Hello, " + name
|[1] pry(main)* end
|=> :hello
|[2] pry(main)> hello("john")
|Hello, john
|=> nil
|[3] pry(main)> "uppercase".up
|.upcase   .upcase!   .upto
|[3] pry(main)> "uppercase".upcase
|=> "UPPERCASE"
|[4] pry(main)> ls
|comple.rb   path.rb
|file        test.rb
|[5] pry(main)> show-method String#upcase
|
|From: string.c (C Method):
|Owner: String
|Visibility: public
|Number of lines: 7
|
|static VALUE
|rb_str_upcase(VALUE str)
|{
```

```
➜ ir  irb
irb(main):001:0> def hello(name)
irb(main):002:1>   puts "Hello, " + name
irb(main):003:1> end
=> :hello
irb(main):004:0> hello("john")
Hello, john
=> nil
irb(main):005:0> "uppercase".upc^C
irb(main):005:0> "uppercase".upcase
=> "UPPERCASE"
irb(main):006:0> `ls`
=> "comple.rb\nfile\npath.rb\ntest.rb\n"
irb(main):007:0>
```

```
|[1] pry(main)> def hello(name)
|[1] pry(main)*   puts "Hello, " + name
|[1] pry(main)* end
|=> :hello
|[2] pry(main)> hello("john")
|Hello, john
|=> nil
|[3] pry(main)> "uppercase".up
|.upcase   .upcase!  .upto
|[3] pry(main)> "uppercase".upcase
|=> "UPPERCASE"
|[4] pry(main)> .ls
|comple.rb       path.rb
|file            test.rb
|[5] pry(main)> show-method String#upcase
|
|From: string.c (C Method):
|Owner: String
|Visibility: public
|Number of lines: 7
|
|static VALUE
|rb_str_upcase(VALUE str)
|{
```

IPython 0.12

IPython 0.12

The major new feature with this release is the IPython Notebook, an interactive Python interface running in the browser. Download it now, or read more about what's new.

– IPython 0.12 Release Note

tebook

Spectrogram

Save

New Open

ownload ipynb

Print

Delete

Code Markdown

Toggle

Above

Up Down

elected All

Autoindent:

nterrupt Restart

kernel upon exit:

Python IPython

lumPy SciPy

MPL SymPy

cted cell

rminal mode

yboard shor

Simple spectral analysis

An illustration of the Discrete Fourier Transform

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = 0, \ldots, N-1$$

using windowing, to reveal the frequency content of a sound signal.
We begin by loading a datafile using SciPy's audio file support:

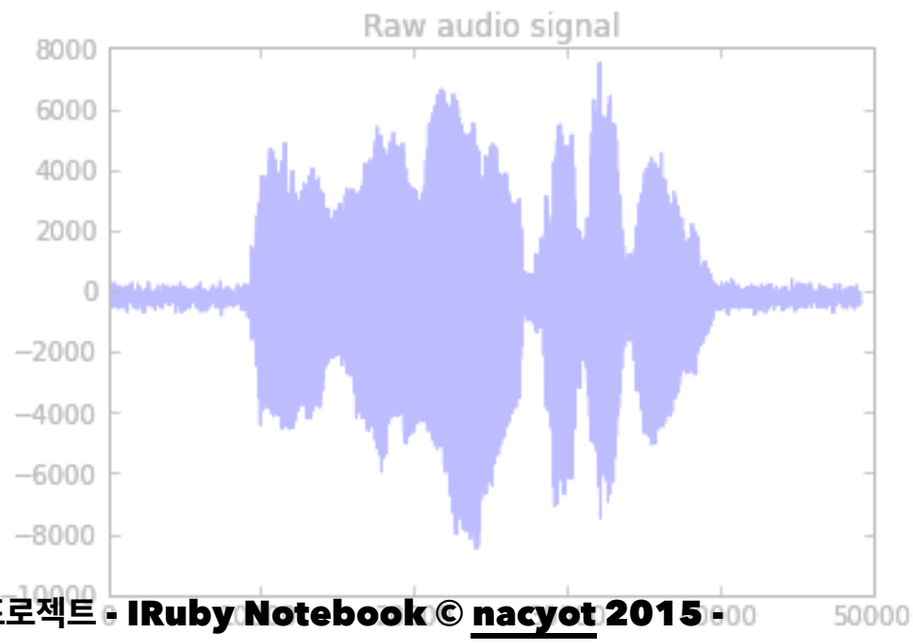
```
In [1]: from scipy.io import wavf
        r..., x = wavf...d('/...e/...p...n...k...e.../...t...wav')
```

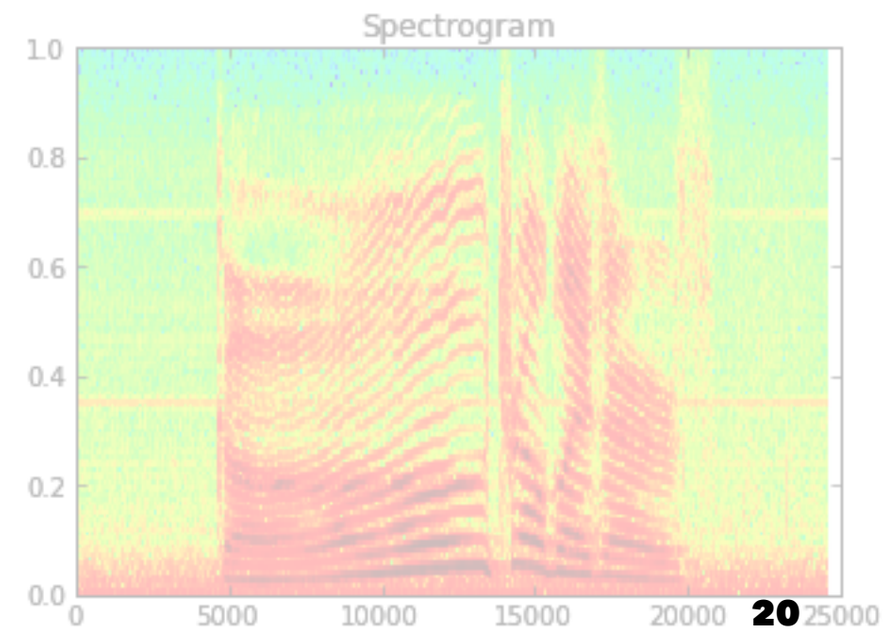
IPython Notebook

2011~

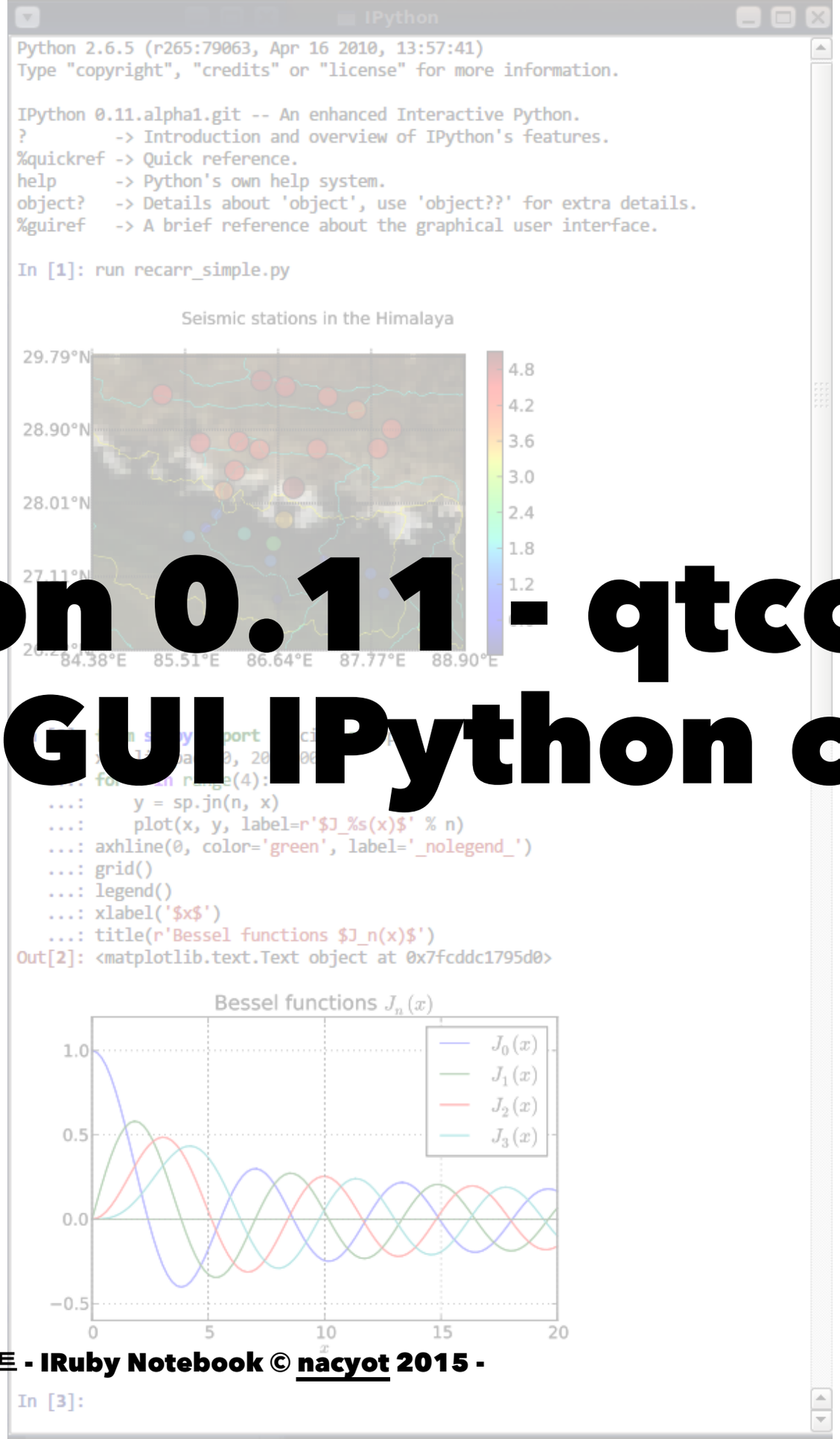
and we can easily view its spectral structure using matplotlib's builtin `specgram` routine:

```
In [3]: fig, (ax1, ax2) = plt.subplots(1, ..., figsize=(12, 4))
        ax1.plot(x); ax1.set_title('Raw audio signal')
        ax2.specgram(x); ax2.set_title('Spectrogram');
```





IPython 0.11



IPython 0.11 - qtconsole Rich GUI IPython client

IPython 0.11 - ZeroMQ

- ZeroMQ 기반 메시지 시스템 도입
- 성능 문제 해결 및 qtconsole 백엔드
- ipython notebook

기존의 REPL

- 해당하는 실행기 언어로 구현
  - lisp -> lisp, irb -> ruby
  - python -> python
- 클라이언트(셸)와 백엔드(실행기)의 강한 결합
  - 분리하기 어려움
  - REPL이 가지는 근본적인 제약

ZeroMQ가 도입된 이유

성능

ZeroMQ가 도입된 이유

ZeroMQ provides us with much tighter control over memory, higher performance, and its communications are impervious to the Python Global Interpreter Lock because they take place in a system-level C++ thread."

– IPython 0.11 Release Note

ZeroMQ 도입에 주목해야 하는 이

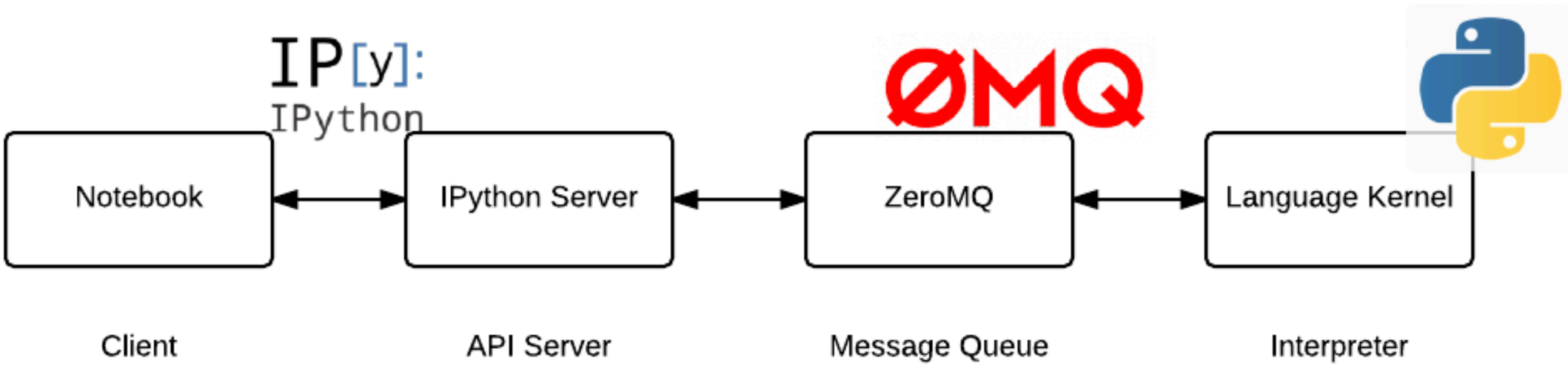
ZeroMQ 이전

단일 프로그램

Interactive Shell & Interpreter

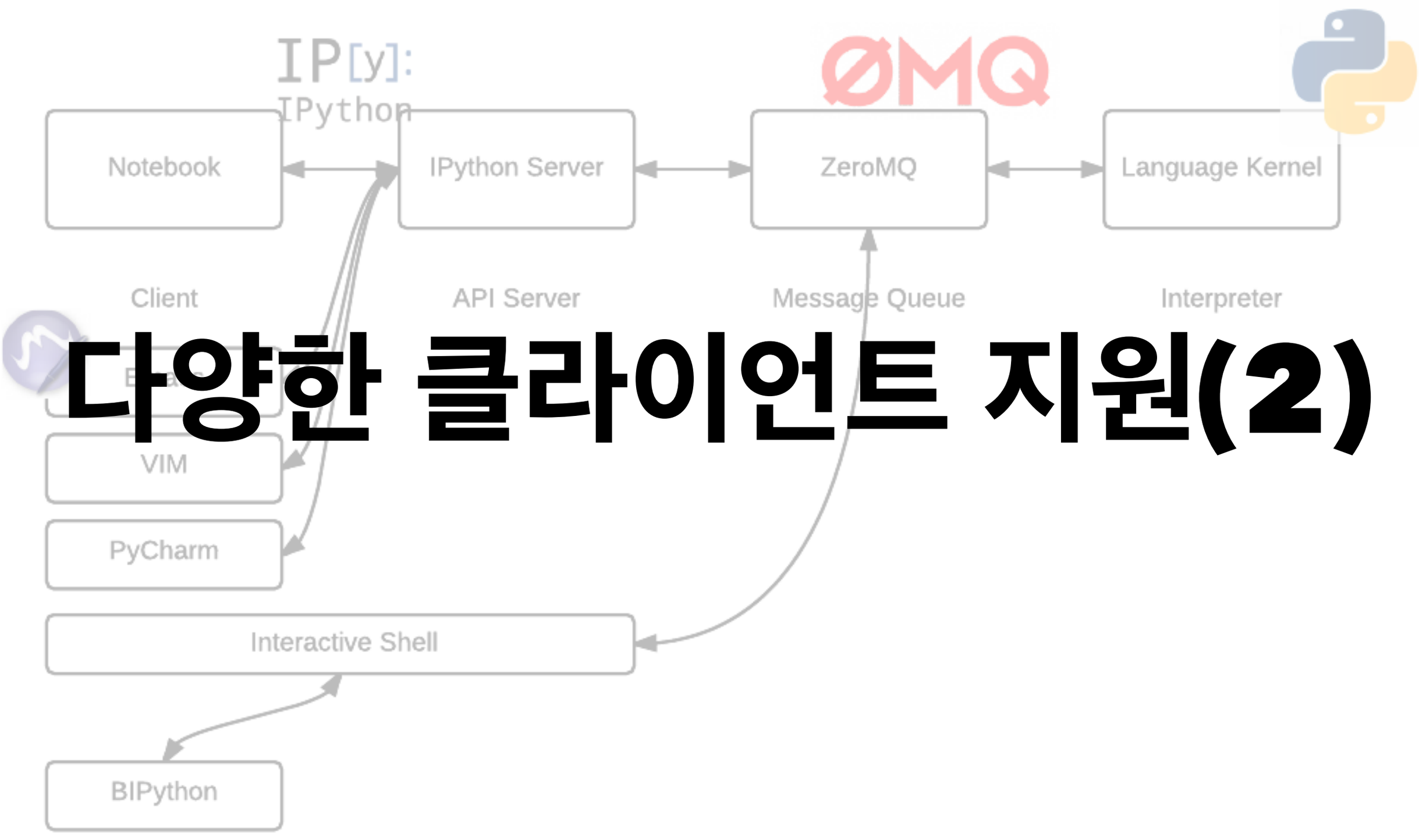
ZeroMQ 이후

클라이언트와 백엔드의 약한 결합

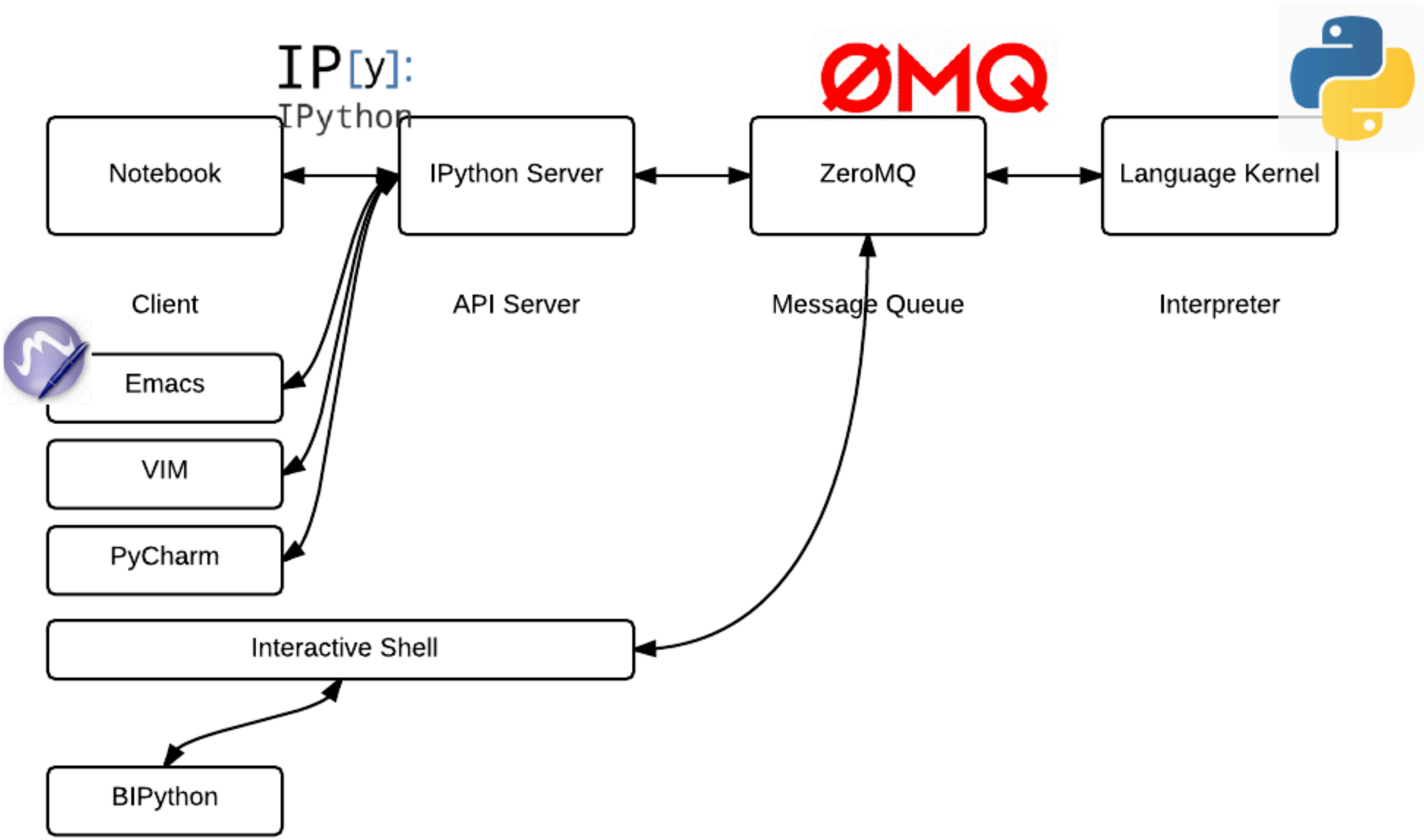


다양한 클라이언트 지원

- Notebook(client) - IPython Server
  - qtconsole
  - IPython Notebook
  - IPython
  - BIPython



다양한 클라이언트 지원(2)


IP[y]:
IPython
ØMQ
Notebook
IPython Server
ZeroMQ
Language Kernel
Client
API Server
Message Queue
Interpreter
Emacs
VIM
PyCharm
Interactive Shell
BIPython

Message Protocol

- Messaging in IPython
- 0.11에서 공개
- 자체 버전을 가지고 있으며 현재 5.0
- ZeroMQ 기반

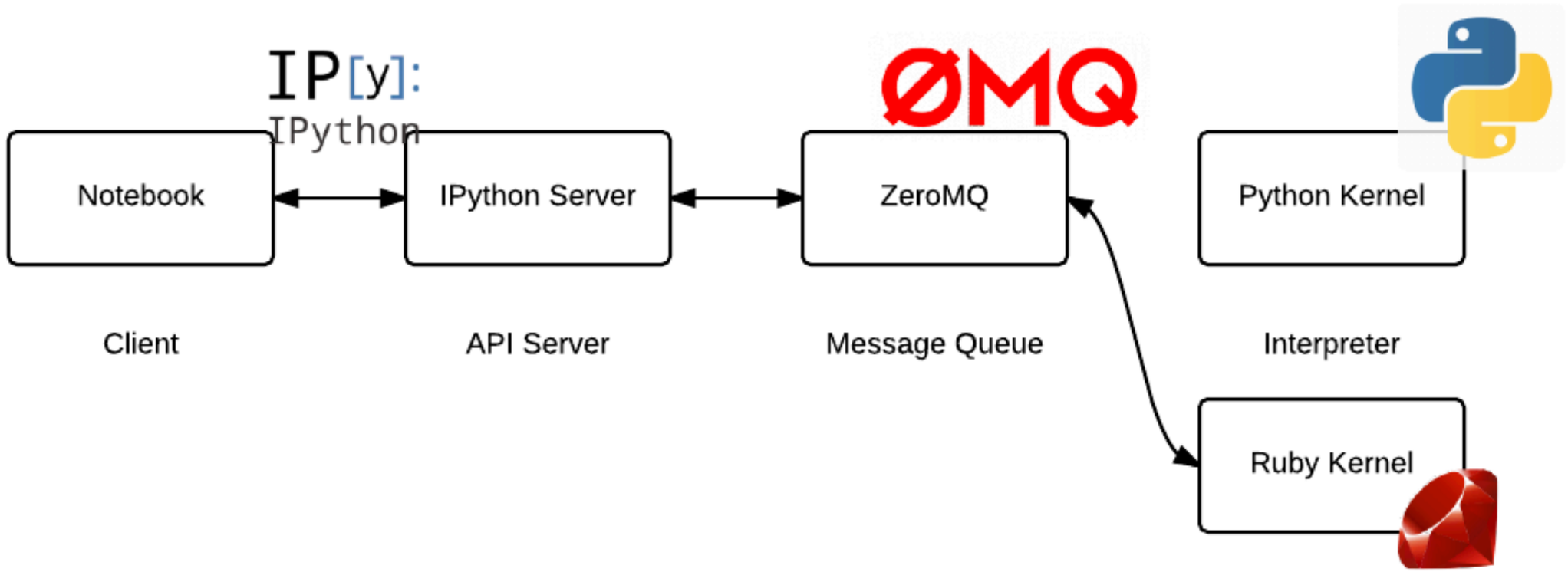
실행기(커널)의 분리

- Message Protocol에 따르는 Kernel 개념 도입
- Python 실행기가 Python Kernel로 분리됨

IRuby의 정의

IRuby = Interactive Ruby?

IRuby = IPython Kernel for Ruby



IPython은 언어에 비종속적인 REPL

- 커널이 분리되면서 다양한 커널이 개발됨
- 기존의 도구/생태계를 그대로 이용 가능

IPython Kernels

- IPython kernels for other languages
  - Julia, Haskell, FSharp, Ruby, Go
  - Scala, Mathics, Aldor, Calico, Erlang
  - Lua, R, OCaml, Forth, Perl, Perl6
  - Octave, Scilab, MathLab, Bash, CSahrp
  - Clojure, Hy, Redis, Javascript, Calysto, ...

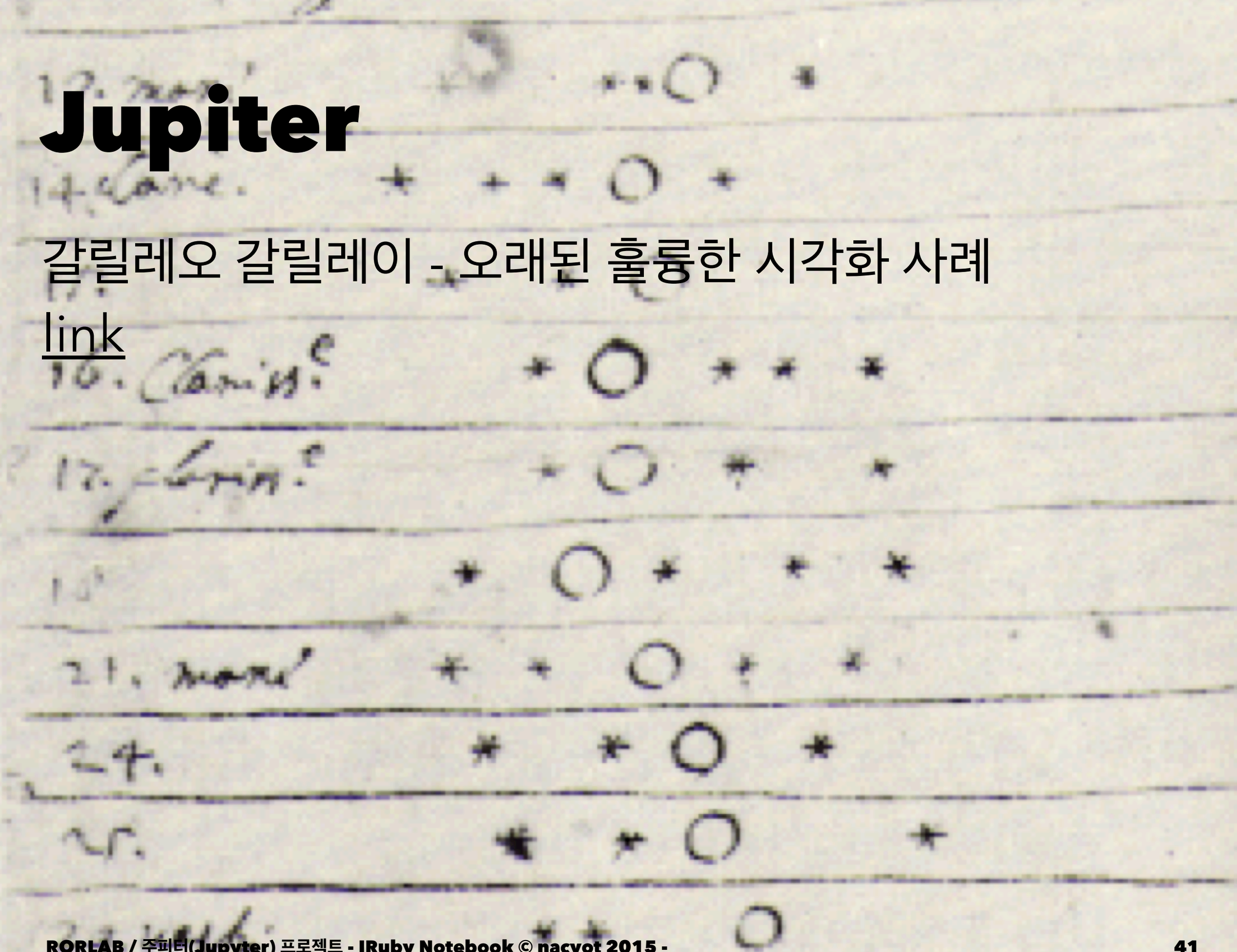
IPython 3.0 = Jupyter

- Project Jupyter
  - 더 이상 Python만을 위한 도구가 아님을 인정
  - 파이썬에 관련된 프로젝트 -> IPython
    - 다언어 지원을 위한 프로젝트 -> Jupyter
- Jupyter Protocol
- Jupyter Notebook(HTML+Javascript 분리)

Jupiter

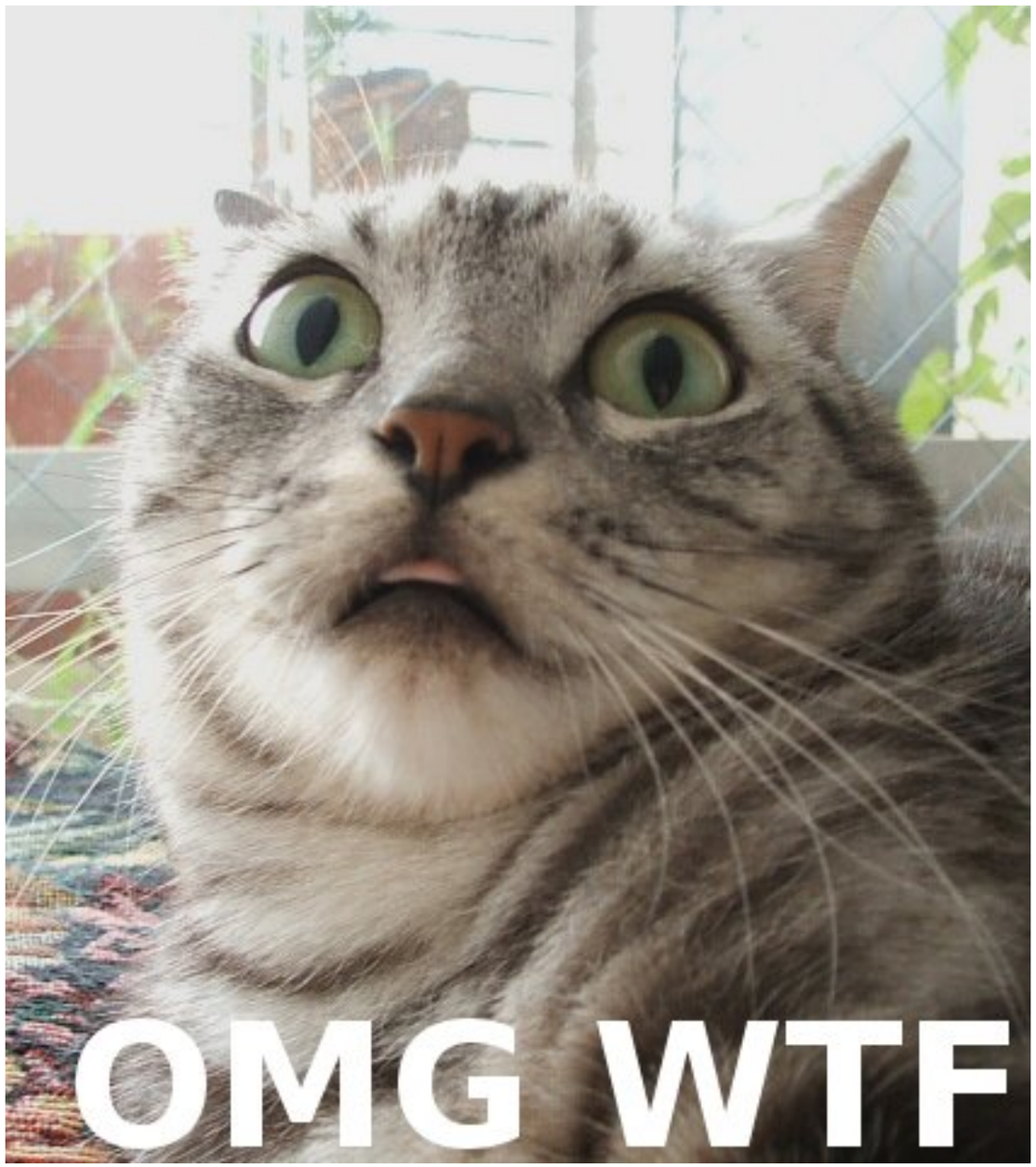
갈릴레오 갈릴레이 - 오래된 훌륭한 시각화 사례

link



Jupyter = Julia + Python + R

Jupyter = Julia + Python + R



Jupyter 이전 - 타언어 커널 사용법

IPython은 커널에 의존적으로 실행

```
$ ipython notebook --kernel <language>
```

Jupyter 이후 - Kernelspec

특정 커널에 비의존적으로 실행

설정 디렉터리 구조

```
$ tree -d -L 1 ~/.ipython
~/.ipython
├── db
├── extensions
├── kernels                 # <- 여기
├── log
├── nbextensions
├── pid
├── profile_default
├── security
├── startup
└── static
```

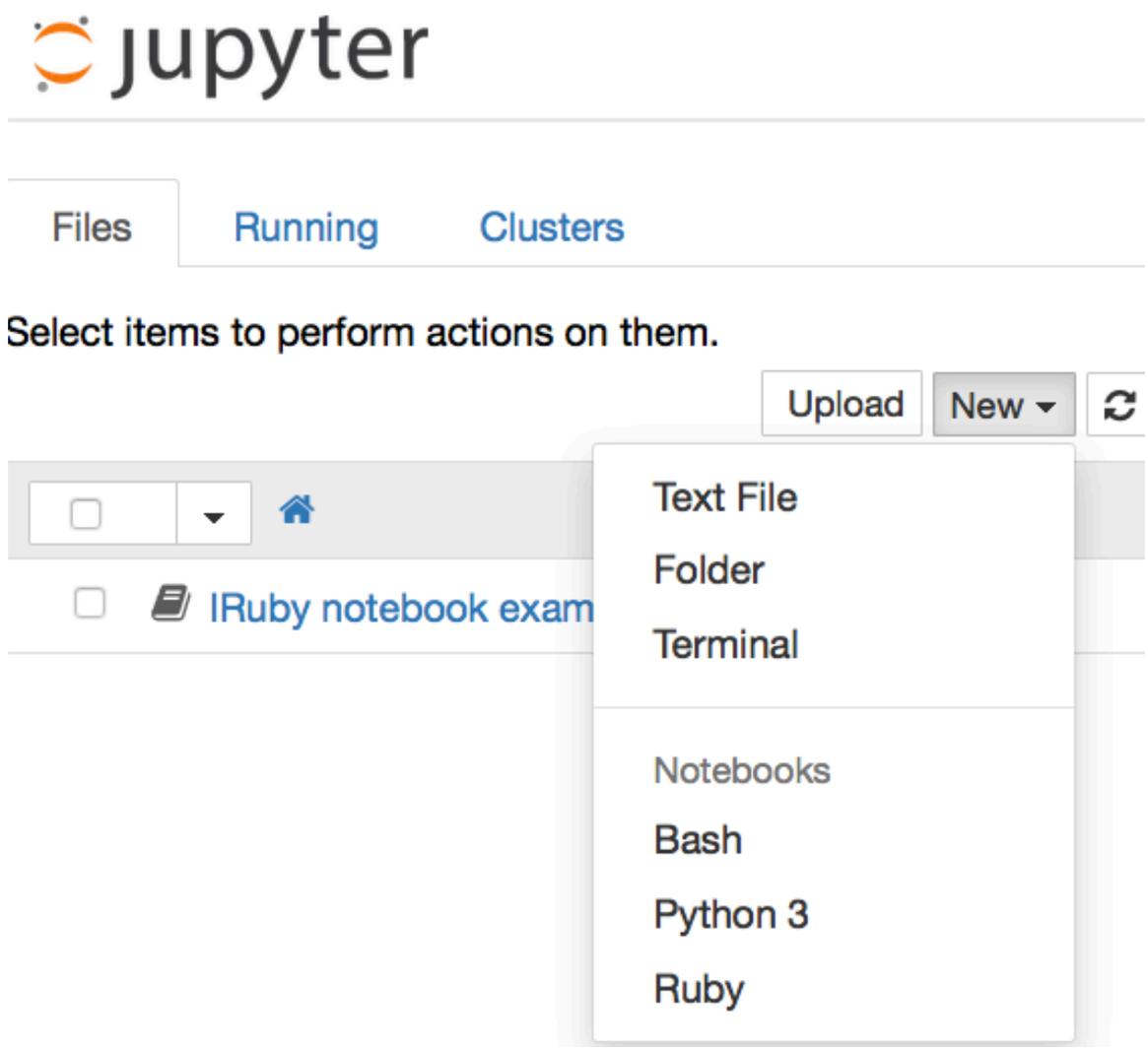
타언어 커널 사용 -> 다언어 커널 사용

IRuby, IBash 설치 이후

```
$ tree -d -L 1 ~/.ipython/kernels
~/.ipython/kernels
├── ruby
└── bash
```

Jupyter Interface

하나의 Jupyter Instance에서 다언어 커널 실행가능



Demo 2

- Ruby
- Haskell
- Bash

IPython Notebook

IPython 0.11~ (2011년~)

What is the big deal about IPython Notebooks?

웹 애플리케이션

- 범용적인 유저 인터페이스
- 비선형적 코드 실행
- CodeMirror 에디터
- Javascript 환경

범용적인 유저 인터페이스

- 웹에서 가능한 모든 것
- HTML, CSS, Image, Canvas, SVG, ...

In [5]:
```
File.open('lib/iruby/static/base/images/ipynblogo.png')
```

Out[5]:

Display

IRuby provides a method to display objects `IRuby.display` and methods to create $\LaTeX$ and HTML representations.

In [6]:
```
IRuby.display '<b style="color:green">Hello, world!</b>', mime: 'text/html'
```

Out[6]: **Hello, world!**

$\LaTeX$ is rendered using MathJax.

In [8]:
```
IRuby.display IRuby.latex <<-'TEX'
\begin{eqnarray}
\nabla \times \vec{\mathbf{B}} -\, \frac1c\, \frac{\partial\vec{\mathbf{E}}}{\partial t} & = \frac
\nabla \cdot \vec{\mathbf{E}} & = 4 \pi \rho \\
\nabla \times \vec{\mathbf{E}}\, +\, \frac1c\, \frac{\partial\vec{\mathbf{B}}}{\partial t} & = \ve
\nabla \cdot \vec{\mathbf{B}} & = 0
\end{eqnarray}
TEX

IRuby.math('F(k) = \int_{-\infty}^{\infty} f(x) e^{2\pi i k} dx')
```

Out[8]:

$$\nabla \times \vec{\mathbf{B}} - \frac{1}{c}\frac{\partial \vec{\mathbf{E}}}{\partial t} = \frac{4\pi}{c}\vec{\mathbf{j}}$$
$$\nabla \cdot \vec{\mathbf{E}} = 4\pi\rho$$

비선형적 코드 실행 (1)

선형적인 실행 - REPL의 본질적인 한계

```
[10] pry(main)> def awesome_upcase_function(str)
[10] pry(main)*   sty = str.upcase
[10] pry(main)*   return str
[10] pry(main)* end
=> :awesome_upcase_function
[11] pry(main)> awesome_upcase_function("super awesome lowercase string")
=> "super awesome lowercase string"
[12] pry(main)> wtf???
[12] pry(main)>
[13] pry(main)>
[14] pry(main)> def awesome_upcase_function(str)
[14] pry(main)*   str = str.upcase
[14] pry(main)*   retunr str
[14] pry(main)* end
=> :awesome_upcase_function
[15] pry(main)> awesome_upcase_function("super awesome lowercase string")
NoMethodError: undefined method `retunr' for main:Object
from (pry):22:in `awesome_upcase_function'
[16] pry(main)> wtf?????????????????????????????????????
[16] pry(main)>
[16] pry(main)>
[16] pry(main)> def awesome_upcase_function(str)
[16] pry(main)*   str = str.upcase
[16] pry(main)*   return str
[16] pry(main)* end
=> :awesome_upcase_function
[17] pry(main)> awesome_upcase_function("super awesome lowercase string")
=> "SUPER AWESOME LOWERCASE STRING"
```

비선형적 코드 실행 (2)

코드는 셀 단위로 편집하고, 실행되고, 재실행

In [4]:
```
def awesome_upcase_function(str)
  str = str.upcase
  return str
end

awesome_upcase_function('super awesome lowercase string')
```

Out[4]: "SUPER AWESOME LOWERCASE STRING"

CodeMirror 에디터

- Support for over 100 languages out of the box
- A powerful, composable language mode system
- Autocompletion (XML)
- Code folding
- Configurable keybindings
- Vim, Emacs, and Sublime Text bindings
- Search and replace interface
- Bracket and tag matching
- Support for split views
- Linter integration
- Mixing font sizes and styles
- Various themes
- Able to resize to fit content
- Inline and block widgets
- Programmable gutters
- Making ranges of text styled, read-only, or atomic
- Bi-directional text support
- Many other methods and addons...

Javascript 환경

Demo 3

- Javascript Magic
- D3 Notebook 예제(시각화 스터디)
- Interactive Widget
- InlineAttachment 예제

여기까지는 프로그래밍 이야기

여기부터는 Notebook 이야기

왜 Jupyter에 주목해야 하는가?

REPL

- 소모성 프로그래밍 환경
- 보통 짧은 코드 테스트용

REPL의 한계를 넘어서

- 기록을 위한 프로그래밍 환경
- 다양한 표현 지원
- 셀 단위의 코드 편집 지원

연습과 기록

Examples are reusable ideas in the form of customizable code snippets; examples can serve as an alternative to fixed, monolithic typologies; examples are a shared extension of memory.

– Eyeo 2013 - For example by Mike Bostock

Demo 4

- naycot/euler_project

코드와 글 사이의 본질적인 문제

코드

IPython code 중 일부 - BSD License

```
def reload(self):
    """Reload the raw data from file or URL."""
    import mimetypes
    if self.embed:
        super(Audio, self).reload()

    if self.filename is not None:
        self.mimetype = mimetypes.guess_type(self.filename)[0]
    elif self.url is not None:
        self.mimetype = mimetypes.guess_type(self.url)[0]
    else:
        self.mimetype = "audio/wav"


def _make_wav(self, data, rate):
    """ Transform a numpy array to a PCM bytestring """
    import struct
    from io import BytesIO
    import wave
```

글

Discourse on Floating Bodies, by Galileo Galilei

A DISCOVRSE

Presented to the Most Serene DON COSIMO II.

GREAT DUKE of *TUSCANY*:

CONCERNING

The Natation of BODIES Upon, or Submersion In, the WATER.

Onsidering (Most Serene Prince) that the publishing this present Treatise, of so different an Argument from that wh many expect, and which according to the intentions I proposed in my * Astronomicall *Adviso*, I should before this t have put forth, might peradventure make some thinke, either that I had wholly relinquished my farther imploym about the new Celestiall Observations, or that, at least, I handled them very remissely; I have judged fit to rende account, aswell of my deferring that, as of my writing, and publishing this treatise.

As to the first, the last discoveries of *Saturn* to be tricorporeall, and of the mutations of Figure in *Venus*, like to t are seen in the Moon, together with the Consequents depending thereupon, have not so much occasioned the demur, as estigation of the times of the Conversions of each of the Four Medicean Planets about *Jupiter*, which I lighted upon in *April* the , 1611, at my being in *Rome*; where, in the end, I assertained my selfe, that the first and neerest to *Jupiter*, moved about 8 *gr.* & 2 ts Sphere in an houre, makeing its whole revolution in one naturall day, and 18 hours, and almost an halfe. The second moves i e 14 *gr.* 13 *min.* or very neer, in an hour, and its compleat conversion is consummate in 3 dayes, 13 hours, and one third reabouts. The third passeth in an hour, 2 *gr.* 6 *min.* little more or less of its Circle, and measures it all in 7 dayes, 4 hours, or very n fourth, and more remote than the rest, goes in one houre, 0 *gr* 54 *min.* and almost an halfe of its Sphere, and finisheth it all i es, and very neer 18 hours. But because the excessive velocity of their returns or restitutions, requires a most scrupulous precisen alculate their places, in times past and future, especially if the time be for many Moneths or Years; I am therefore forced, with o servations, and more exact than the former, and in times more remote from one another, to correct the Tables of such Motions, it them even to the shortest moment: for such exactnesse my first Observations suffice not; not only in regard of the short interval e, but because I had not as then found out a way to measure the distances between the said Planets by any Instrument: I Obser h Intervals with simple relation to the Diameter of the Body of *Jupiter*; taken, as we have said, by the eye, the which, though t hit not errors of above 1 Minute, yet they suffice not for the determination of the exact greatness of the Spheres of these Stars. v that I have hit upon a way of taking such measures without failing, scarce in a very few Seconds, I will continue the observatio

코드와 글 ≒ 물과 기름

- 각자 고유한 맥락을 가짐
  - 전혀 다른 방식으로 쓰여짐
  - 도구로 처리할 수 있는 부분이 다름
- 문법 분석과 맞춤법/오타
  - Syntax 하이라이팅
  - 80자 제한과 문단 개념

글 안에 포함된 "죽어있는" 코드

책에 실린 코드

1. Array 객체의 스택 메소드 ⚑

x `Array#<<` 메소드는 인수를 하나만 받습니다만 `Array#push`는 여러개의 인수를 받을 수 있습니다. 또한 `Array#pop`는 한 번에 여러개의 값을 pop 할 수 있습니다. `Array#unshift` `Array#shift`도 마찬가지입니다.

```
stack = []
stack.push 1, 2, 3 # => [1, 2, 3]
stack.pop 2 # => [2, 3]
stack # => [1]
stack.unshift 4, 5, 6 # => [4, 5, 6, 1]
stack.shift 3 # => [4, 5, 6]
stack # => [1]
```

코드와 글의 고유한 맥락에 대한 고민들

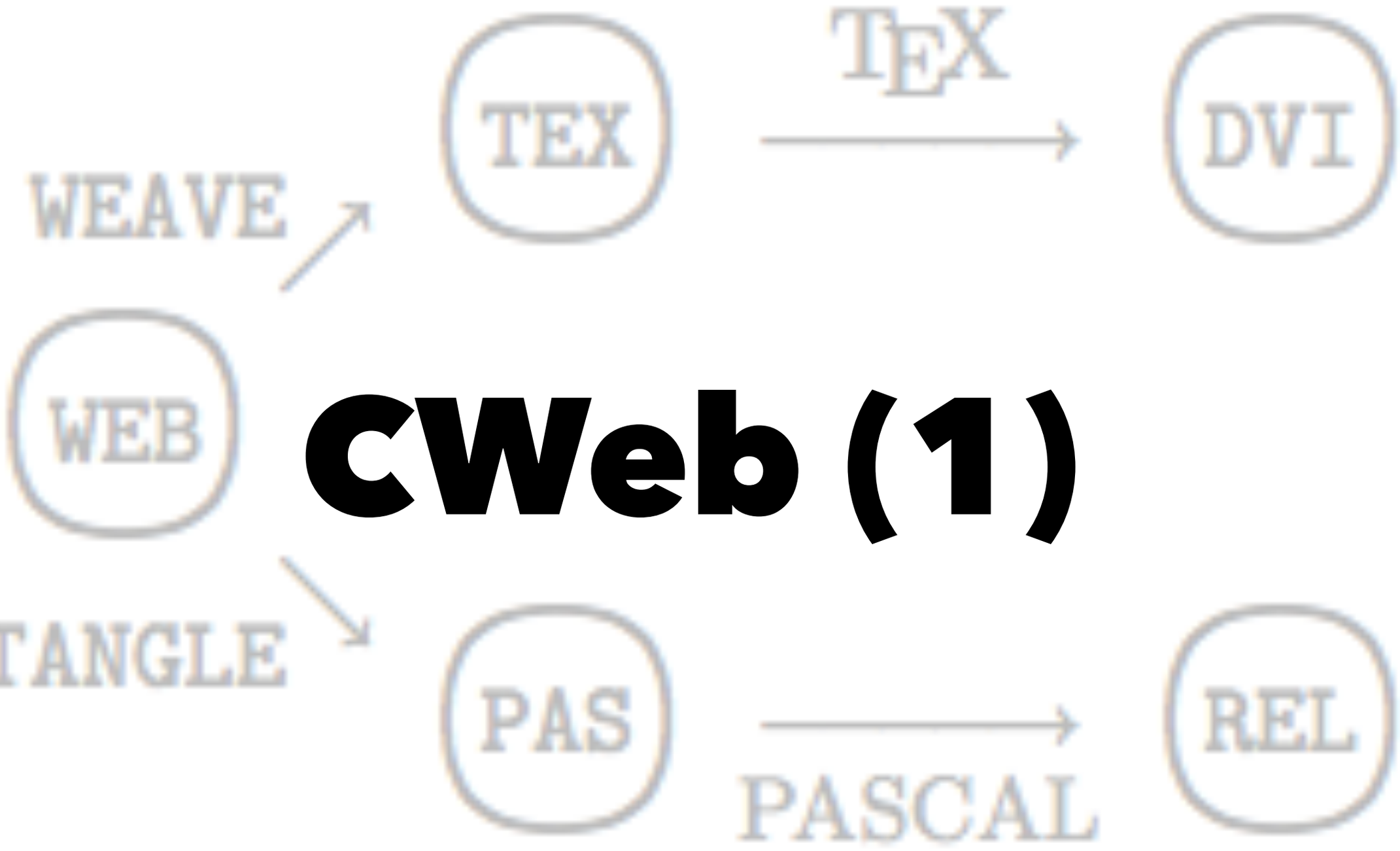
- Donal E. Knuth - Literate Programming
- Alan Kay - Active Essays
- Fernando Perez - Data-driven Journalism

Literate Programming(문학적 프로그래밍)

Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.

– Donald E. Knuth

같은 평면 위에 올려진 코드와 문서



CWeb (1)

. Dual usage of a WEB file.

CWeb (2)

구조적 문서화를 위한 CWEB 시스템

```
@ 프로시져 |prime_the_change_buffer|는 계속되는 매칭 과정에서 다음
매칭 동작을 위해서 |change_buffer|를 세팅한다. 수정 파일에 있는 빈 줄은 매칭에
쓰이지 않기 때문에, |(change_limit==change_buffer && !changing)|와
같은 검사를 해야한다. 즉, 수정 파일을 다 읽어들였는지 확인해야
한다. |changing|이 1일 때만 호출된다. 따라서 에러
메시지들은 정확히 예상한 줄에 나올 것이다.

@c
void
prime_the_change_buffer()
{
  change_limit=change_buffer; /* 이 값은 수정 파일이 끝났는지를 체크할 때 사용된다. */
  @<수정 파일에서 주석문을 무시하고 건너뛰어라; 파일의 끝이면 |return|@>;
  @<빈 줄이 아닐 때까지 건너뛰어라; 파일의 끝이면 |return|@>;
  @<|buffer|와 |limit|의 내용을 |change_buffer|와 |change_limit|으로 옮겨라@>;
}

@ 수정 파일에서 \.{@@x}로 시작하는 줄들을 찾는 과정에서, \.{@@y},
\.{@@z}, \.{@@i}로 시작하지 않으면서 \.{@@}로 시작하는 줄이 나와도 된다.
(만약 그러한 명령어로 시작된다면, 수정 파일이 잘못되었다는 것을 뜻한다.)

@<수정 파일에서 주석문을 무시하고 건너뛰어라...@>=
while(1) {
  change_line++;
  if (!input_ln(change_file)) return;
  if (limit<buffer+2) continue;
  if (buffer[0]!='@@') continue;
  if (xisupper(buffer[1])) buffer[1]=tolower(buffer[1]);
  if (buffer[1]=='x') break;
  if (buffer[1]=='y' || buffer[1]=='z' || buffer[1]=='i') {
    loc=buffer+2;
    err_print("! Missing @@x in change file");
@.Missing @@x...@>
  }
}
```

"살아있는" 코드 안에 포함된 글

Sphinx (1)

ansible-modules-core/cloud/docker/docker.py

```
23
24  DOCUMENTATION = '''
25  ---
26  module: docker
27  version_added: "1.4"
28  short_description: manage docker containers
29  description:
30    - Manage the life cycle of docker containers.
31  options:
32    count:
33      description:
34        - Number of matching containers that should be in the desired state.
35      default: 1
36    image:
37      description:
38        - Container image used to match and launch containers.
39      required: true
```

Sphinx (2)

Ansible Documentation - docker

Docs » docker - manage docker containers

docker - manage docker containers

- Synopsis
- Options
- Examples
- This is a Core Module

Synopsis

New in version 1.4.

Manage the life cycle of docker containers.

Options

parameter	required	default	choices	comments
command	no			Command used to match and launch containers.
count	no	1		Number of matching

글 안에 포함된 "살아있는" 코드

knitr (1)

from **future** import dream
knitr를 이용한 워드프레스 포스팅하기

아래는 최근 보고 있는 R Graphics Cookbook에서 본 코드를 넣어본 것이다. 자동으로 그래프와 출력물들이 블로그 포스팅에 포함이 되게 되어 데이터 기반 블로그를 작성할때 매우 편하게 작업할 수 있다.

코드 예시

```
# from R Graphics Cookbook
library(gcookbook)
library(ggplot2)
library(plyr)

#데이터 헤드
head(uspopage)
#요약 통계량
summary(uspopage)

ggplot(uspopage, aes(x=Year, y=Thousands, fill=AgeGroup, order=desc(AgeGro
geom_area(colour=NA, alpha=.4) +
scale_fill_brewer(palette="Blues") +
geom_line(position="stack", size=.2)
```

knitr (2)

from **future** import dream
knitr를 이용한 워드프레스 포스팅하기

아래는 최근 보고 있는 R Graphics Cookbook에서 본 코드를 넣어본 것이다.
자동으로 그래프와 출력물들이 블로그 포스팅에 포함이 되게 되어 데이터 기반 블로그를 작성할때 매우 편하게 작업할 수 있다.

코드 예시

```
# from R Graphics Cookbook
library(gcookbook)
library(ggplot2)
library(plyr)

# 데이터 헤드
head(uspopage)
```

```
##   Year AgeGroup Thousands
## 1 1900       <5      9181
## 2 1900     5-14     16966
## 3 1900    15-24     14951
## 4 1900    25-34     12161
## 5 1900    35-44      9273
## 6 1900    45-54      6437
```

```
# 요약 통계량
summary(uspopage)
```

```
##       Year         AgeGroup     Thousands
##  Min.   :1900   <5      :103   Min.   : 3099
##  1st Qu.:1925   5-14    :103   1st Qu.:13557
##  Median :1951   15-24   :103   Median :19759
##  Mean   :1951   25-34   :103   Mean   :20900
##  3rd Qu.:1977   35-44   :103   3rd Qu.:24410
##  Max.   :2002   45-54   :103   Max.   :45154
##                 (Other):206
```

d3 도서(o'reilly)

O'Reilly Atlas + jsbin

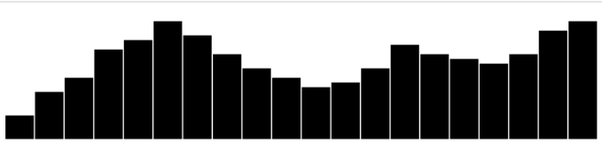


Figure 6-25. Growing bigger from above

The working code for our growing-down-from-above, SVG bar chart is in *17_making_a_bar_chart_heights.html*.

Try It Now!

Try adding or deleting data points from `dataset` at left and see how the width of the bars adjusts at right. Then try adjusting the values of the data points and see the changes reflected in the height of the bars.

Open in new window

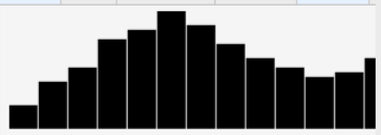
JS Bin Save

HTML CSS JavaScript Console Output

HTML

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>D3: A true bar chart
with SVG rects</title>
      <script
type="text/javascript"
src="http://d3js.org
/d3.v3.min.js"></script>
    <style type="text/css">
      /* No style rules here yet
*/
    </style>
  </head>
  <body>
    <script
type="text/javascript">

      //Width and height
```



Auto-run JS Run with JS

Active Essays

An "Active Essay" is a new kind of literacy, combining a written essay, live simulations, and the programs that make them work in order to provide a deep explanation of a dynamic system. The reader works directly with multiple ways of representing the concepts under discussion. By "playing with" the simulations and code, the reader gets some hands-on experience with the topic.

– Alan Kay

Steven Wittens' Presentation

- Making WebGL Dance
- Source Code

Setosa blog

- Markov Chains

Jiongster

- ..., Why React is Awesome
- Presenting The Most Over-Engineered Blog Ever

Awesome!

- 표현 도구로서의 자바스크립트
- 하지만, 하나의 웹사이트라고 봐야...
- 저작 환경 = 그냥 프로그래밍

그리고, Jupyter

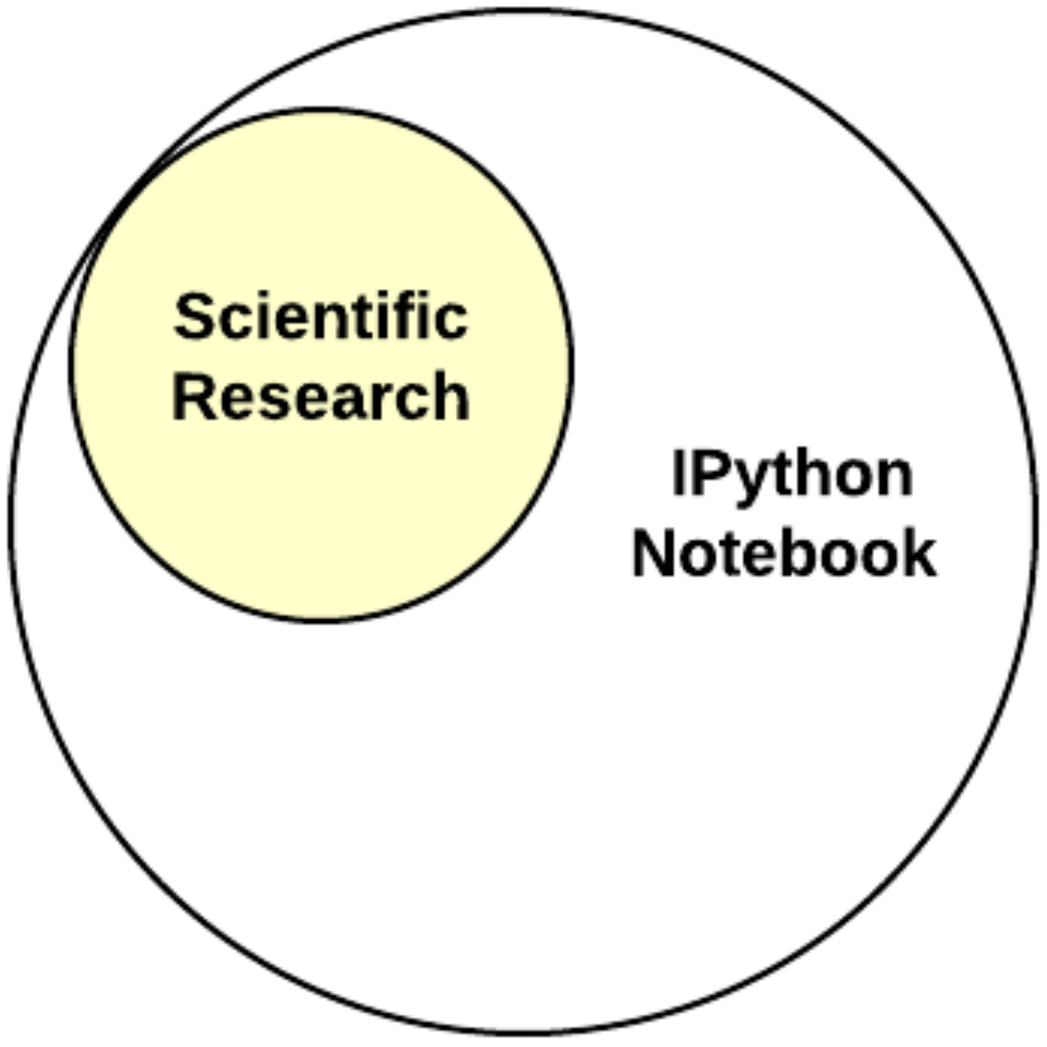
"코드"와 "연구"의 만남

현존하는 가장 범용적인 *Scientific Research* 환경.

"글"과 "코드"의 고유한 맥락

현존하는 가장 범용적인 *Active Essays* 저작 환경.

Scientific Research



Data-driven Journalism

Our job with IPython is to think deeply about questions regarding the intersection of computing, data and science, but it's clear to me at this point that we can contribute in contexts beyond pure scientific research. I hope we'll be able to provide folks who have a direct intersection with the public, such as journalists, with tools that help a more informed and productive debate.

– Fernando Perez

Reproducible Notebook

- 코드를 실행해보는 것에 그치지 않음
  - 저장하고, 게시하고, 공유하도록 도와줌
  - ipython 환경이 있다면 실행해보는 것도 가능
- HTML 등 다른 포맷으로 출력 기능 제공
  - nbviewer
  - ipynb 뷰어 서버

IPython Notebook으로 블로그하기

- Fernando Perez
  Blogging with the IPython notebook
- Pythonic Perambulations
- Box and Whisker
  IPython Notebook으로 블로깅하기

사례

루비의 꽃, 열거자 Enumerable 모듈
루비(Ruby) 테스트 프레임워크 RSpec 2.14 매쳐

Apr 19 2014
each enumerable map ruby 루비 열거자 표준 라이브러리 프로그래밍 언어

루비의 꽃, 열거자 Enumerable 모듈

프로그래밍을 배우면 피해갈 수 없는 부분 중 하나가 바로 제어 추상화입니다. 그 중에서도 반복문은 특히 많이 사용되는데, 재미있는 건 루비에서는 다른 언어에서 많이 사용되는 while이나 for 같은 문법을 잘 사용하지 않는다는 점입니다. 이러한 변수 재대입에 의존한 반복문들을 사용하기보다는 컬렉션의 요소 하나하나를 블록에 넘겨 평가하는 `each`와 같은 열거자(Enumerable) 메서드가 주로 사용됩니다. 이러한 컬렉션 확장 메서들은 처음 사용할 때는 낯설게 느껴질 지도 모르지만, 사실은 컬렉션 없는 반복문이야 말로 특수한 경우이므로 루비의 접근이 합리적이라는 걸 금방 깨닫게됩니다. 나아가 Enumerable은 단순히 `each` 메서드만 제공하는 게 아닙니다. 다양한 열거자 메서드를 통해 루비에서 컬렉션을 좀 더 자유자재로 다룰 수 있습니다. 이 글에서는 Enumerable 모듈에 포함된 다양한 열거자 메서드들을 소개합니다.

개요

앞서 이야기했듯이 루비에서는 아래 스타일의 반복문을 일부 지원하지만 별로 사용하지 않습니다

83번째 입력:

```
i = 0
while i < 5 do
  puts i
  i += 1
end
```

.ipynb 빌드

helper/markdown_helper.rb

```
def render_ipynb(filename)
  source = "./source/iruby/#{filename}.ipynb"
  output = "./source/iruby/#{filename}"
  cmd = "ipython nbconvert --to html --template basic #{source} --output #{output}"
  system(cmd)
end
```

Demo 5

- 루비의 꽃, 열거자 Enumerable 모듈 실행하기

Thank you!

@nacyo_t