Through the lens of Haskell

Exploring new ideas for library design

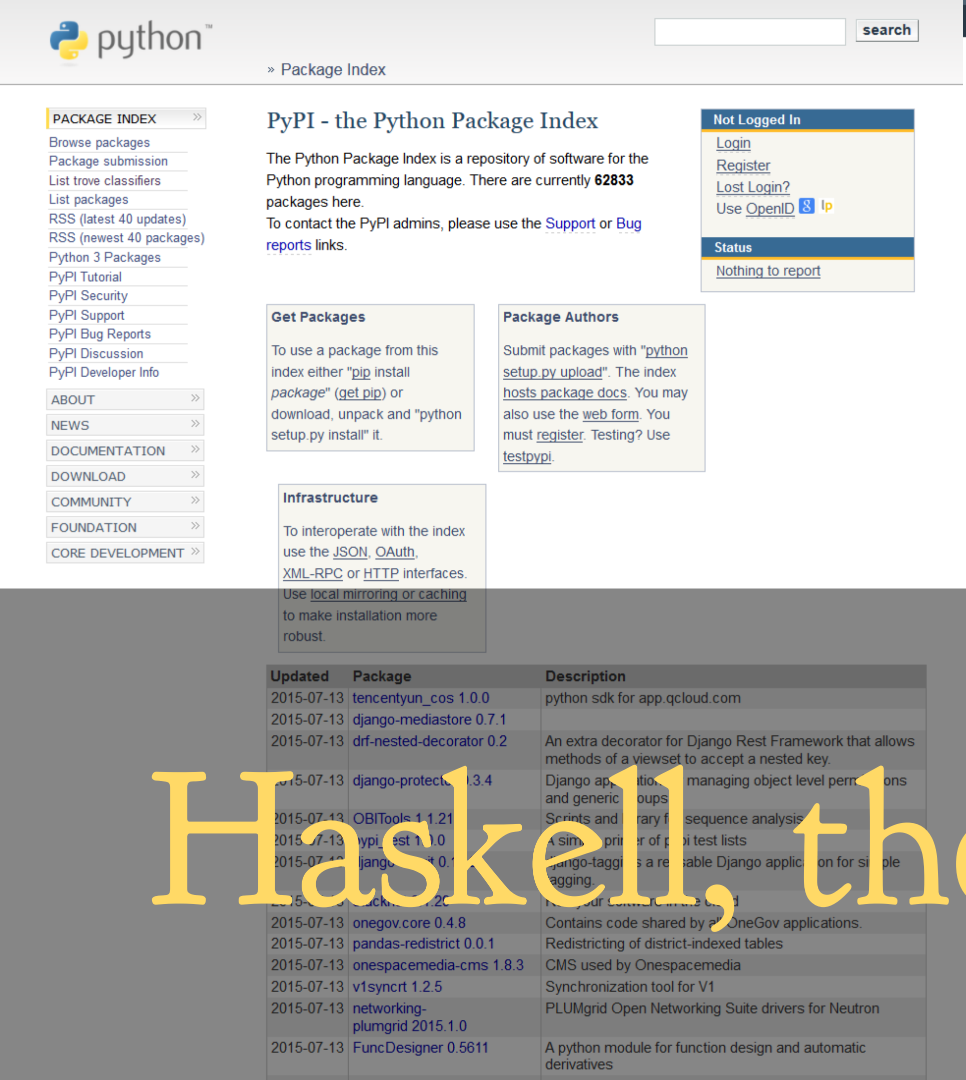
@georgesdubus

What Python can learn from Haskell

Bob Ippolito (@etrepum)
EuroPython Berlin
21 July 2014
bob.ippoli.to/python-haskell-ep2014

Haskell, the language

Haskell, the ecosystem



PyPI - the Python Package Index

The Python Package Index is a repository of software for the Python programming language. There are currently **62833** packages here.
To contact the PyPI admins, please use the Support or Bug reports links.

Get Packages

To use a package from this index either "pip install *package*" (get pip) or download, unpack and "python setup.py install" it.

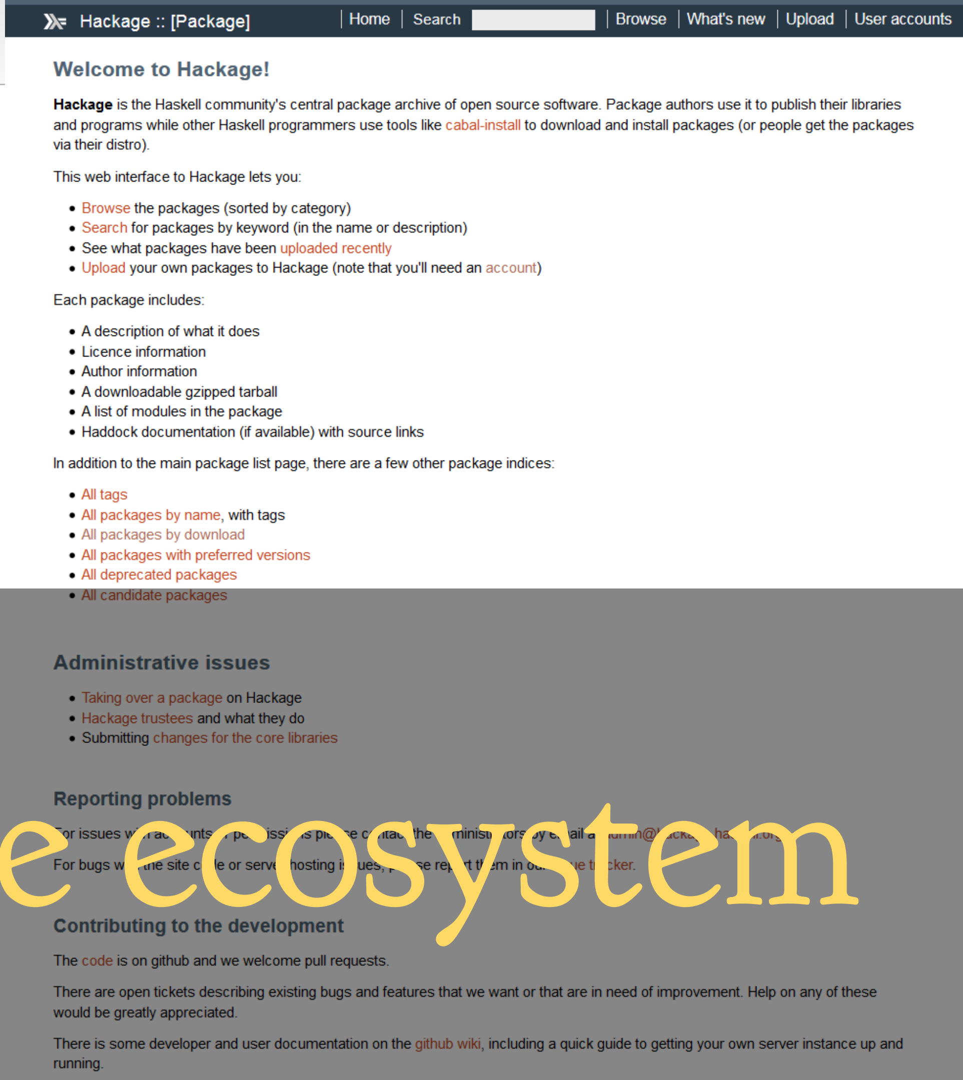
Package Authors

Submit packages with "python setup.py upload". The index hosts package docs. You may also use the web form. You must register. Testing? Use testpypi.

Infrastructure

To interoperate with the index use the JSON, OAuth, XML-RPC or HTTP interfaces. Use local mirroring or caching to make installation more robust.

Updated	Package	Description
2015-07-13	tencentyun_cos 1.0.0	python sdk for app.qcloud.com
2015-07-13	django-mediastore 0.7.1	
2015-07-13	drf-nested-decorator 0.2	An extra decorator for Django Rest Framework that allows methods of a viewset to accept a nested key.
[illegible]15-07-13	django-protect[illegible].3.4	Django ap[illegible]tion[illegible] managing object level perm[illegible]ons and generic [illegible]oups
2015-07-13	OBITools 1.1.21	Scripts and [illegible]rary f[illegible] sequence analysis
2015-[illegible]7-13	[illegible]ypi_[illegible]est 1[illegible].0	A sim[illegible] prin[illegible]er of p[illegible]i test lists
2015-07-[illegible]	[illegible]jango[illegible]t 0.1[illegible]	[illegible]jango-taggi[illegible]s a re[illegible]sable Django applic[illegible]on for si[illegible]ple tagging.
[illegible]	[illegible]	[illegible]
2015-07-13	onegov.core 0.4.8	Contains code shared by al[illegible] OneGov applications.
2015-07-13	pandas-redistrict 0.0.1	Redistricting of district-indexed tables
2015-07-13	onespacemedia-cms 1.8.3	CMS used by Onespacemedia
2015-07-13	v1syncrt 1.2.5	Synchronization tool for V1
2015-07-13	networking-plumgrid 2015.1.0	PLUMgrid Open Networking Suite drivers for Neutron
2015-07-13	FuncDesigner 0.5611	A python module for function design and automatic derivatives



Welcome to Hackage!

Hackage is the Haskell community's central package archive of open source software. Package authors use it to publish their libraries and programs while other Haskell programmers use tools like cabal-install to download and install packages (or people get the packages via their distro).

This web interface to Hackage lets you:

- Browse the packages (sorted by category)
- Search for packages by keyword (in the name or description)
- See what packages have been uploaded recently
- Upload your own packages to Hackage (note that you'll need an account)

Each package includes:

- A description of what it does
- Licence information
- Author information
- A downloadable gzipped tarball
- A list of modules in the package
- Haddock documentation (if available) with source links

In addition to the main package list page, there are a few other package indices:

- All tags
- All packages by name, with tags
- All packages by download
- All packages with preferred versions
- All deprecated packages
- All candidate packages

Administrative issues

- Taking over a package on Hackage
- Hackage trustees and what they do
- Submitting changes for the core libraries

Reporting problems

For issues w[illegible] ac[illegible]unts [illegible] pe[illegible]issi[illegible]s ple[illegible]se c[illegible]ntac[illegible] the [illegible]minist[illegible]tors by e[illegible]ail a[illegible]dmin@[illegible]ha[illegible].org

For bugs w[illegible] the site c[illegible]le or serve[illegible] hosting i[illegible]ues, [illegible]se rep[illegible]t th[illegible]em in ou[illegible]ue tr[illegible]cker.

Contributing to the development

The code is on github and we welcome pull requests.

There are open tickets describing existing bugs and features that we want or that are in need of improvement. Help on any of these would be greatly appreciated.

There is some developer and user documentation on the github wiki, including a quick guide to getting your own server instance up and running.

Design space

There should be one
— and preferably only one —
obvious way to do it. (Python)

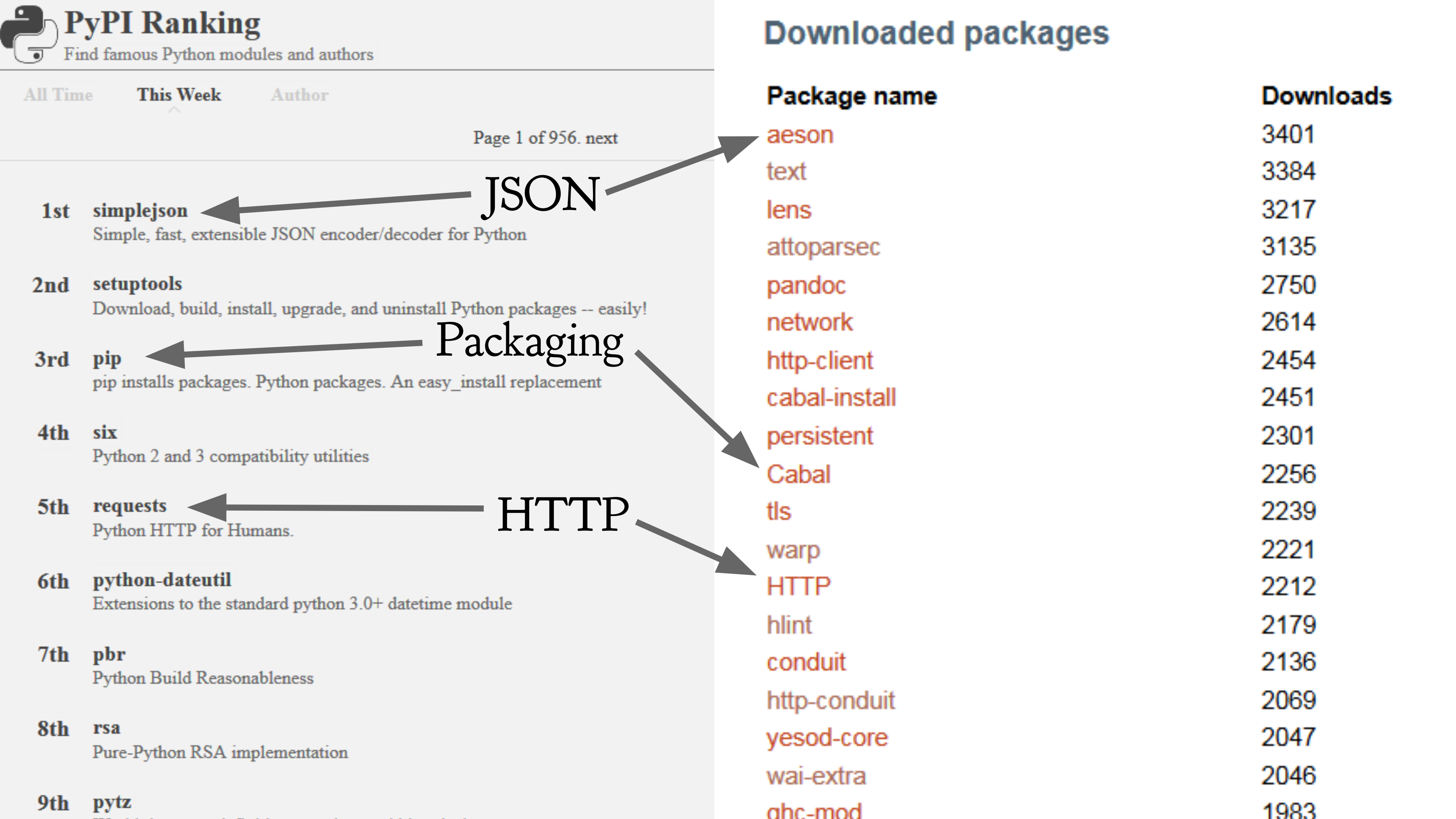
There should be one
— and preferably only one —
obvious way to do it. (Python)

Let's keep looking for it! (Haskell)

Python For Humans

Kenneth Reitz

Some Haskell libraries



PyPI Ranking
Find famous Python modules and authors

All Time | This Week | Author

Page 1 of 956. next

1st **simplejson**
Simple, fast, extensible JSON encoder/decoder for Python

2nd **setuptools**
Download, build, install, upgrade, and uninstall Python packages -- easily!

3rd **pip**
pip installs packages. Python packages. An easy_install replacement

4th **six**
Python 2 and 3 compatibility utilities

5th **requests**
Python HTTP for Humans.

6th **python-dateutil**
Extensions to the standard python 3.0+ datetime module

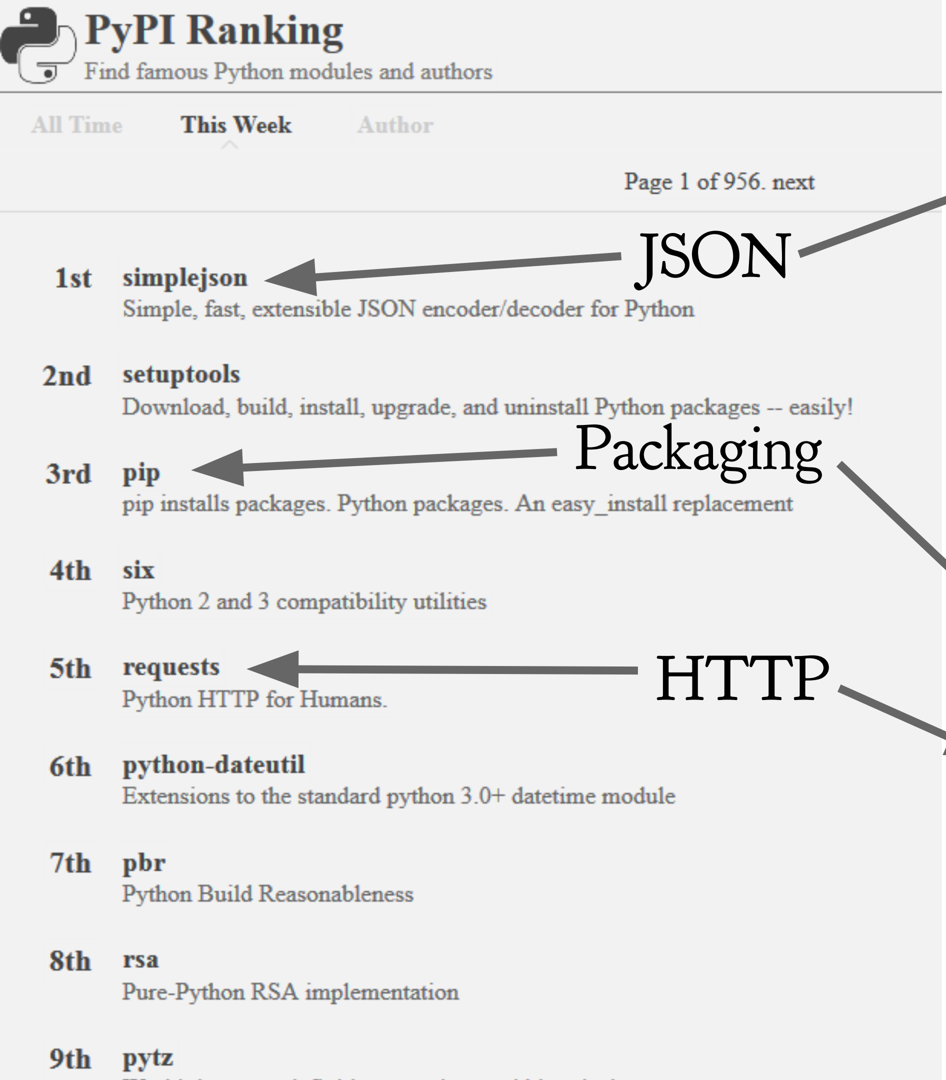
7th **pbr**
Python Build Reasonableness

8th **rsa**
Pure-Python RSA implementation

9th **pytz**

Downloaded packages

Package name	Downloads
aeson	3401
text	3384
lens	3217
attoparsec	3135
pandoc	2750
network	2614
http-client	2454
cabal-install	2451
persistent	2301
Cabal	2256
tls	2239
warp	2221
HTTP	2212
hlint	2179
conduit	2136
http-conduit	2069
yesod-core	2047
wai-extra	2046
ghc-mod	1983

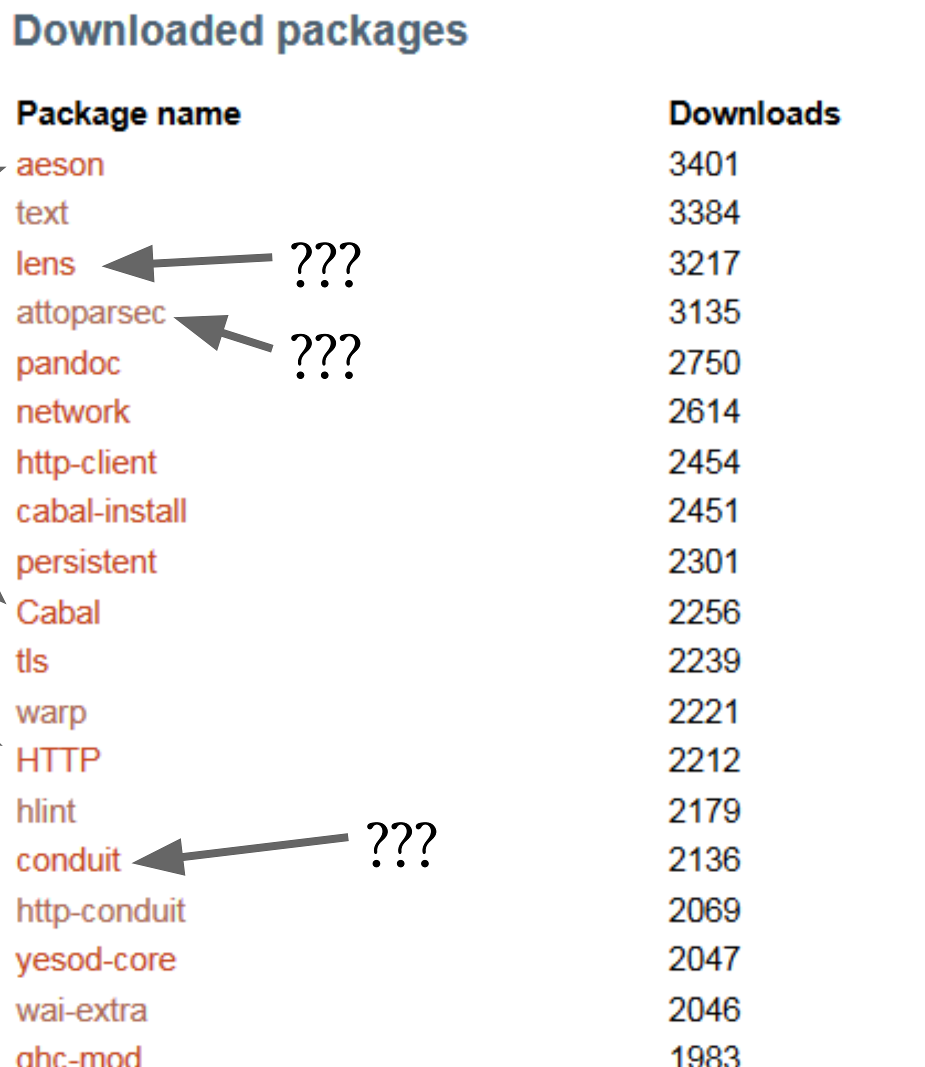


PyPI Ranking
Find famous Python modules and authors

All Time | **This Week** | Author

Page 1 of 956. next

1st	**simplejson** Simple, fast, extensible JSON encoder/decoder for Python
2nd	**setuptools** Download, build, install, upgrade, and uninstall Python packages -- easily!
3rd	**pip** pip installs packages. Python packages. An easy_install replacement
4th	**six** Python 2 and 3 compatibility utilities
5th	**requests** Python HTTP for Humans.
6th	**python-dateutil** Extensions to the standard python 3.0+ datetime module
7th	**pbr** Python Build Reasonableness
8th	**rsa** Pure-Python RSA implementation
9th	**pytz**



Downloaded packages

Package name	Downloads
aeson	3401
text	3384
lens	3217
attoparsec	3135
pandoc	2750
network	2614
http-client	2454
cabal-install	2451
persistent	2301
Cabal	2256
tls	2239
warp	2221
HTTP	2212
hlint	2179
conduit	2136
http-conduit	2069
yesod-core	2047
wai-extra	2046
ghc-mod	1983

JSON

???

???

Packaging

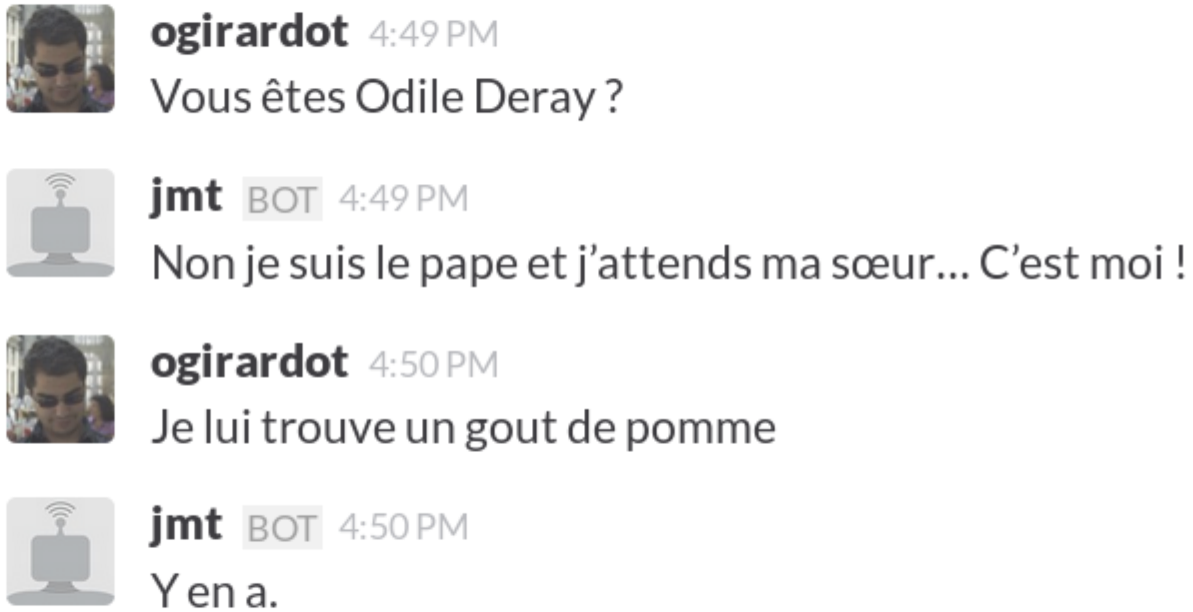
HTTP

???

Some Haskell libraries

Part 1: attoparsec

“Real life” use case :



ogirardot 4:49 PM
Vous êtes Odile Deray ?

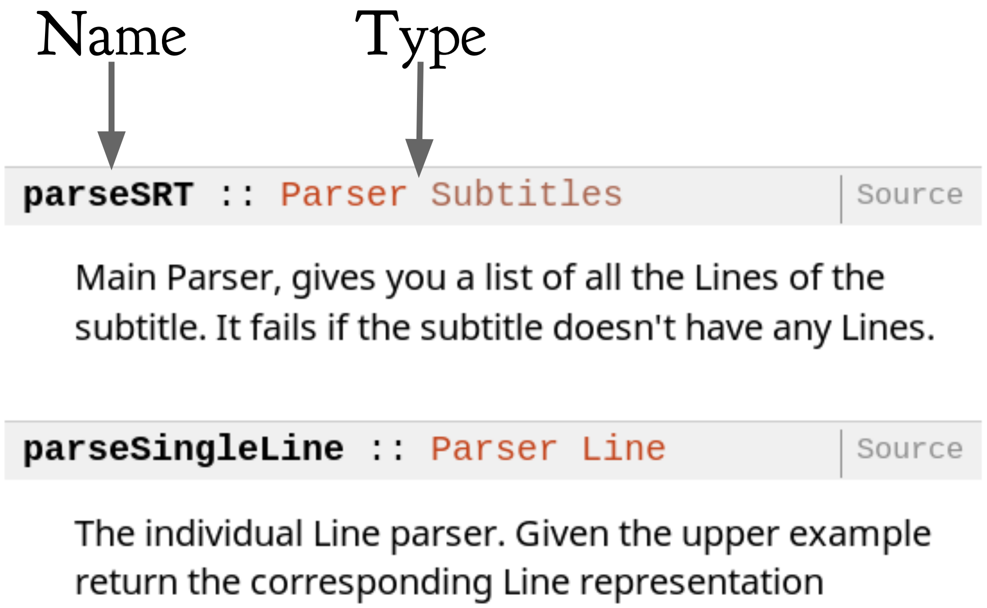
jmt BOT 4:49 PM
Non je suis le pape et j’attends ma sœur… C’est moi !

ogirardot 4:50 PM
Je lui trouve un gout de pomme

jmt BOT 4:50 PM
Y en a.

A slack bot that answers movie quotes

subtitleParser

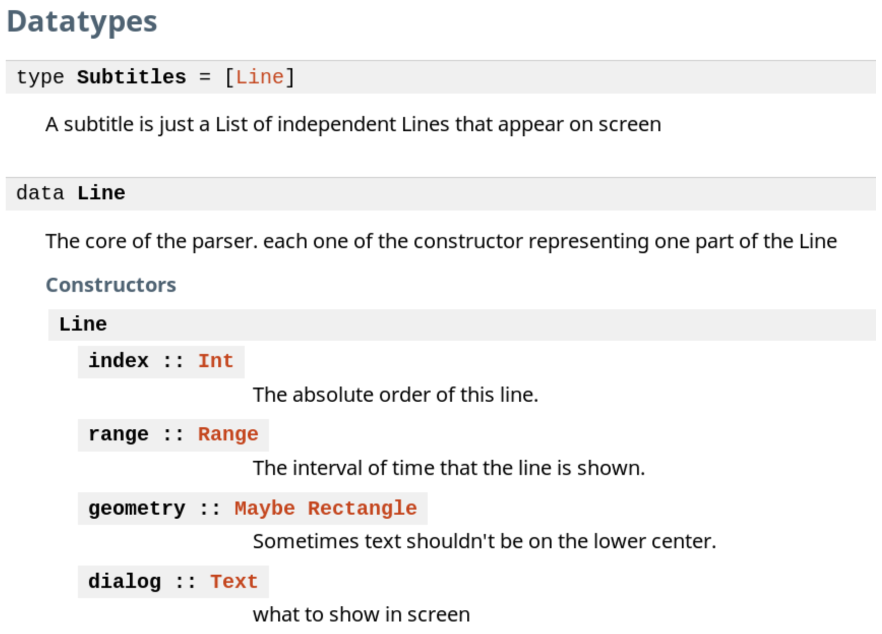


```
parseSRT :: Parser Subtitles                Source
```

Main Parser, gives you a list of all the Lines of the subtitle. It fails if the subtitle doesn't have any Lines.

```
parseSingleLine :: Parser Line              Source
```

The individual Line parser. Given the upper example return the corresponding Line representation



Datatypes

```
type Subtitles = [Line]
```

A subtitle is just a List of independent Lines that appear on screen

```
data Line
```

The core of the parser. each one of the constructor representing one part of the Line

Constructors

```
Line
  index :: Int
```

The absolute order of this line.

```
  range :: Range
```

The interval of time that the line is shown.

```
  geometry :: Maybe Rectangle
```

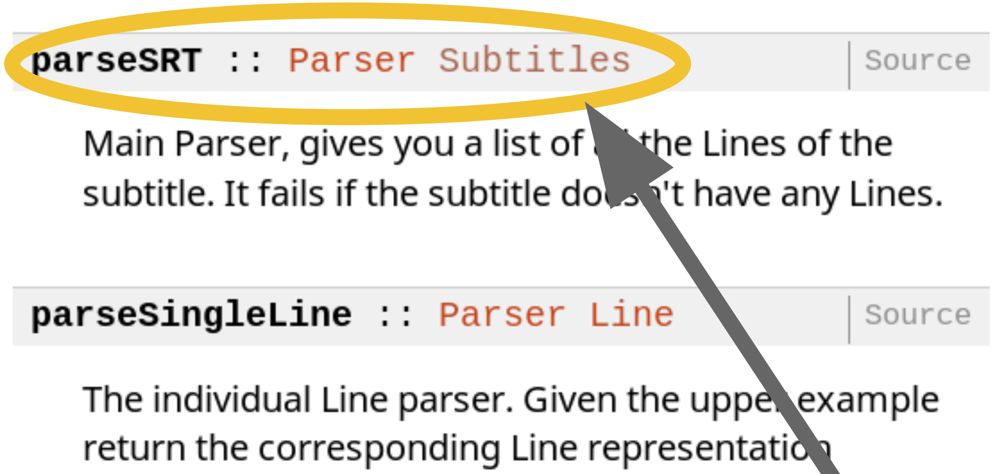
Sometimes text shouldn't be on the lower center.

```
  dialog :: Text
```

what to show in screen

Full package definition

subtitleParser

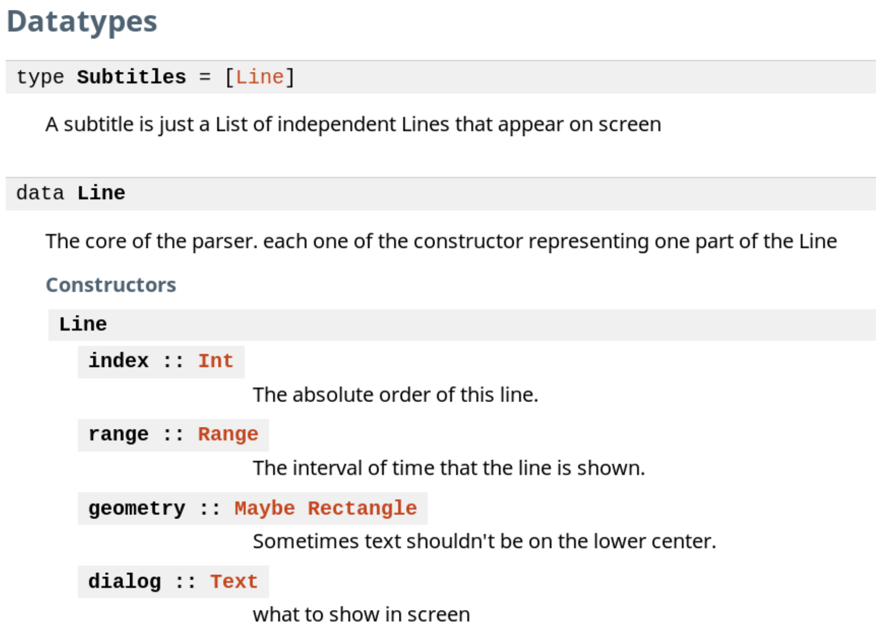


```
parseSRT :: Parser Subtitles                Source
```

Main Parser, gives you a list of all the Lines of the subtitle. It fails if the subtitle doesn't have any Lines.

```
parseSingleLine :: Parser Line              Source
```

The individual Line parser. Given the upper example return the corresponding Line representation



Datatypes

```
type Subtitles = [Line]
```

A subtitle is just a List of independent Lines that appear on screen

```
data Line
```

The core of the parser. each one of the constructor representing one part of the Line

Constructors

```
Line
  index :: Int
```
The absolute order of this line.

```
  range :: Range
```
The interval of time that the line is shown.

```
  geometry :: Maybe Rectangle
```
Sometimes text shouldn't be on the lower center.

```
  dialog :: Text
```
what to show in screen

All I need

attoparsec

All I need to know :

```
parseOnly :: Parser Subtitles -> ByteString -> Either ErrorMessage Subtitles
```

attoparsec

All I need to know :

```
parseOnly :: Parser Subtitles -> ByteString -> Either ErrorMessage Subtitles


parseOnly :: Parser a         -> ByteString -> Either ErrorMessage a
```

attoparsec

Or : incremental parsing

```
parse :: Parser a -> ByteString -> Result a

feed :: Result a -> ByteString -> Result a
```

(Result can be Partial, Failed or Done)

attoparsec

Part of a bigger parser

```
many :: Parser a -> Parser [a]

or :: Parser a -> Parser b -> Parser (Either a b)
```

Parsers everywhere

```
parseCSV      :: Parser CSV                  in attoparsec-csv
json          :: Parser JSONValue            in aeson
crontab       :: Parser Crontab              in cron
emailAddress  :: Parser String               in email-header
toml          :: Parser TOMLValue            in toml
...
```

A good library simplifies the implementation

A good library simplifies the interface

General solution
Specific building blocks

attoparsec

General solution

Specific building blocks

all parsers

Some Haskell libraries

Part 2: conduit

Conduit

Streaming library

Producers
Consumers
Conduits that both consume and produce

Lot of libraries

```
sourceSocket socket =$= ungzip =$= sinkFile "/tmp/output"
```

producer from Data.Conduit.Network

conduit from Data.Conduit.Zlib

consumer from Data.Conduit.Binary

conduit + attoparsec = 😍

Parser ➡ Conduit

```
sourceFile “something.srt” =$= conduitParser parseSubtitleLine =$= ircConsumer
```

Parser of Subtitle Lines

High-performance subtitles streaming for free !

conduit

General solution

Specific building blocks

all conduits

Some Haskell libraries

Part 3: lens

Data manipulation

```
BlogPost { title   = “Made-up examples considered harmful”
         , author = Person {name=“Alice”}
         , comments = [
             Comment { author = “Bob”
                     , content = “Great insight!”
                     }
           , Comment { author = “Carol”
                     , content = “I completely disagree”
                     }
           ]
         }
```

Getters

Lens

```
>>> view title blogpost
“Made-up examples considered harmful”
```

Getters

Lens

```
>>> view title blogpost
“Made-up examples considered harmful”
```

Lens Lens

```
>>> view (author . name) blogpost
"Alice"
```

Getters

Lens

```
>>> view title blogpost
“Made-up examples considered harmful”
```

Lens Lens

```
>>> view (author . name) blogpost
"Alice"
```

Setters

```
>>> set (speaker . name) “Alicia” blogpost
BlogPost { title   = “Made-up examples considered harmful”
         , author = “Alicia”
         , ...
         }
```

Getters/setters with multiple values ?!?

Lens Traversal Lens

```
>>> toListOf (comments . each . author) blogpost
[“Bob”, “Carol”]
```

Getter / setter pairs are values

```
>>> let commentContents = comments . each . content

>>> toListOf commentContents blogpost
[“Great insight!”, “I completely disagree”]

>>> set commentContents “Blah blah blah” blogpost
BlogPost { comments = [
             Comment { author = “Bob”
                     , content = “Blah blah blah”
                     }
           , Comment { author = “Carol”
                     , content = “Blah blah blah”
                     }
           ]
         , ...
         }
```

Libraries provide lenses: JSON

```
[{“id”: “1”, “name”: “georges”},
 {“id”: “2”, “name”: “lucie”}]

>>> input & (values . key “name”) %~ capitalize
[{“id”: “1”, “name”: “Georges”},
 {“id”: “2”, “name”: “Lucie”}]
```

Libraries provide lenses: HTML

```
titles = allNamed (only "h2") . contents
```

Traversal into all tags with a given name

Their content

lens

General solution

Specific building blocks

all lenses

“Borrowing” ideas

Python ➡ Haskell

wreq = requests + lens

```
ghci> import Network.Wreq
ghci> r <- get "http://httpbin.org/get"
```

```
ghci> import Control.Lens
ghci> r ^. responseHeader "Content-Type"
"application/json"
```

```
ghci> import Data.Aeson.Lens
ghci> r ^.. responseBody . key "items" . values .
            key "owner" . key "login" . _String
["steffi2392","rmies","Spacejoker","walpen",{-...-}
```

Haskell ➡ Python

hypothesis

```
@given(text())
def test_decode_inverts_encode(s):
    assert decode(encode(s)) == s
```

Conclusion

Explore the design space

Explore the design space

Factorize library interfaces

Explore the design space

Factorize library interfaces

Bonus : DIY conclusion