Through the lens of Haskell

Exploring new ideas for library design
Haskell, the language
Haskell, the ecosystem
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.  

Let’s keep looking for it!
Some Haskell libraries
PyPI Ranking
Find famous Python modules and authors

Downloaded packages

<table>
<thead>
<tr>
<th>Package name</th>
<th>Downloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>aeson</td>
<td>3401</td>
</tr>
<tr>
<td>text</td>
<td>3384</td>
</tr>
<tr>
<td>lens</td>
<td>3217</td>
</tr>
<tr>
<td>attoparsec</td>
<td>3135</td>
</tr>
<tr>
<td>pandoc</td>
<td>2750</td>
</tr>
<tr>
<td>network</td>
<td>2614</td>
</tr>
<tr>
<td>http-client</td>
<td>2454</td>
</tr>
<tr>
<td>cabal-install</td>
<td>2451</td>
</tr>
<tr>
<td>persistent</td>
<td>2301</td>
</tr>
<tr>
<td>Cabal</td>
<td>2256</td>
</tr>
<tr>
<td>tls</td>
<td>2239</td>
</tr>
<tr>
<td>warp</td>
<td>2212</td>
</tr>
<tr>
<td>HTTP</td>
<td>2212</td>
</tr>
<tr>
<td>http-conduit</td>
<td>2212</td>
</tr>
<tr>
<td>yesod-conduit</td>
<td>2047</td>
</tr>
<tr>
<td>wai-extra</td>
<td>2047</td>
</tr>
<tr>
<td>GHC-mod</td>
<td>1983</td>
</tr>
</tbody>
</table>

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

JSON
Packaging
HTTP
<table>
<thead>
<tr>
<th>Package name</th>
<th>Downloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>aeson</td>
<td>3401</td>
</tr>
<tr>
<td>text</td>
<td>3384</td>
</tr>
<tr>
<td>lens</td>
<td>???</td>
</tr>
<tr>
<td>attoparsec</td>
<td>???</td>
</tr>
<tr>
<td>pandoc</td>
<td>3217</td>
</tr>
<tr>
<td>network</td>
<td>3135</td>
</tr>
<tr>
<td>http-client</td>
<td>2750</td>
</tr>
<tr>
<td>cabal-install</td>
<td>2614</td>
</tr>
<tr>
<td>persistent</td>
<td>2454</td>
</tr>
<tr>
<td>Cabal</td>
<td>2451</td>
</tr>
<tr>
<td>tls</td>
<td>2301</td>
</tr>
<tr>
<td>warp</td>
<td>2256</td>
</tr>
<tr>
<td>HTTP</td>
<td>2454</td>
</tr>
<tr>
<td>hlint</td>
<td>2239</td>
</tr>
<tr>
<td>conduit</td>
<td>2221</td>
</tr>
<tr>
<td>http-conduit</td>
<td>2212</td>
</tr>
<tr>
<td>yesod-core</td>
<td>2179</td>
</tr>
<tr>
<td>wai-extra</td>
<td>2136</td>
</tr>
<tr>
<td>ghc-mod</td>
<td>2069</td>
</tr>
<tr>
<td>http-conduit</td>
<td>2047</td>
</tr>
<tr>
<td>yesod-core</td>
<td>2046</td>
</tr>
<tr>
<td>wai-extra</td>
<td>1983</td>
</tr>
</tbody>
</table>

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

---

**PyPI Ranking**

Find famous Python modules and authors

<table>
<thead>
<tr>
<th>All Time</th>
<th>This Week</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Page 1 of 956.

Next
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>parseSRT</td>
<td>Parser Subtitles</td>
</tr>
<tr>
<td>parseSingleLine</td>
<td>Parser Line</td>
</tr>
</tbody>
</table>

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

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

Datatypes

```haskell
type subtitles = [Line]
```

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

```haskell
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

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

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
  range :: Range
  geometry :: Maybe Rectangle
  dialog :: Text

  The absolute order of this line.
  The interval of time that the line is shown.
  Sometimes text shouldn't be on the lower center.
  what to show in screen
allI need to know:

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

All I need to know:

\[
\text{parseOnly} :: \text{Parser Subtitles} \rightarrow \text{ByteString} \rightarrow \text{Either ErrorMessage Subtitles}
\]

\[
\text{parseOnly} :: \text{Parser a} \rightarrow \text{ByteString} \rightarrow \text{Either ErrorMessage a}
\]
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Parser Type</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>parseCSV</td>
<td>Parser CSV</td>
<td>in attoparsec-csv</td>
</tr>
<tr>
<td>json</td>
<td>Parser JSONValue</td>
<td>in aeson</td>
</tr>
<tr>
<td>crontab</td>
<td>Parser Crontab</td>
<td>in cron</td>
</tr>
<tr>
<td>emailAddress</td>
<td>Parser String</td>
<td>in email-header</td>
</tr>
<tr>
<td>toml</td>
<td>Parser TOMLValue</td>
<td>in toml</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A good library simplifies the implementation
A good library simplifies the interface
General solution
Specific building blocks
Some Haskell libraries

Part 2: conduit
Conduit

Streaming library

Producers
Consumers
Conduits that both consume and produce
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!
General solution
Specific building blocks

conduit

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

```python
>>> view title blogpost
"Made-up examples considered harmful"

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

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

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

Setters

```haskell
>>> set (speaker . name) “Alicia” blogpost
BlogPost { title = “Made-up examples considered harmful”
, author = “Alicia”
, ...
}
Getters/setters with multiple values ?!?

```python
>>> toListOfl(comments . each . author) blogpost
["Bob", "Carol"]
```
Getter / setter pairs are values

```javascript
>>> 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

```json
[{
  "id": "1",
  "name": "georges"
},
{
  "id": "2",
  "name": "lucie"
}]
```

```plaintext
>>> 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
General solution
Specific building blocks

lens
all lenses
“Borrowing” ideas
Python ➔ Haskell

\[ wreq = \text{requests} + \text{lens} \]

```
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
["stefi2392","rmies","Spacejoker","walpen",{-...-}]
```
Haskell ➔ Python
hypothesis

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