When RegEx is Not Enough

Nati Cohen (@nocoot)

PyCon Israel 2016

Nati Cohen (@nocoot)

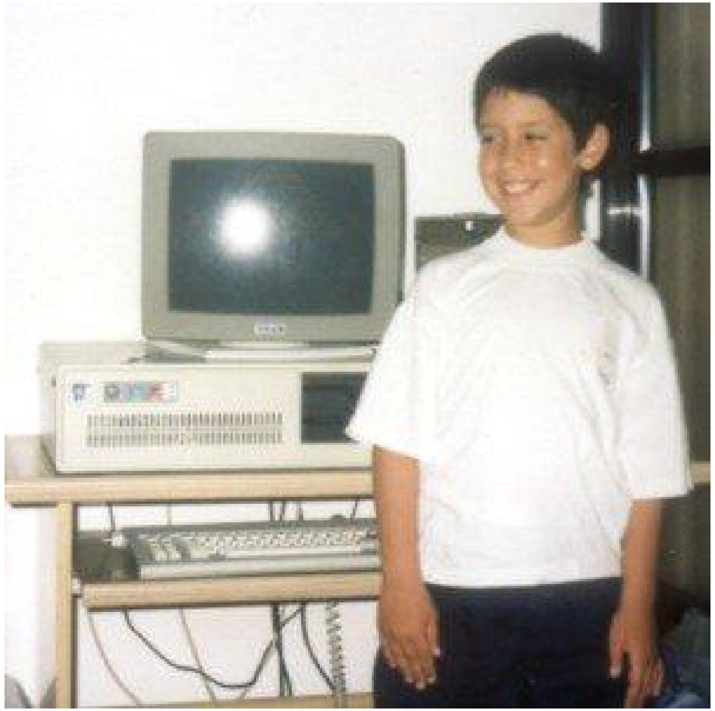
Production Engineer @ SimilarWeb

CS MSc Student @ IDC Herzliya

Co-organizing:

- OpsTalk Meetup Group
- Statscraft Conference

[CALL FOR SPEAKERS]



The Task

“We need you to read our app’s configuration, and do <STUFF> with it”

Too easy, right?

```
---
import ConfigParser

config = ConfigParser.RawConfigParser()
config.read('app.cfg')
# do <STUFF>
```

Oh, and it's not INI

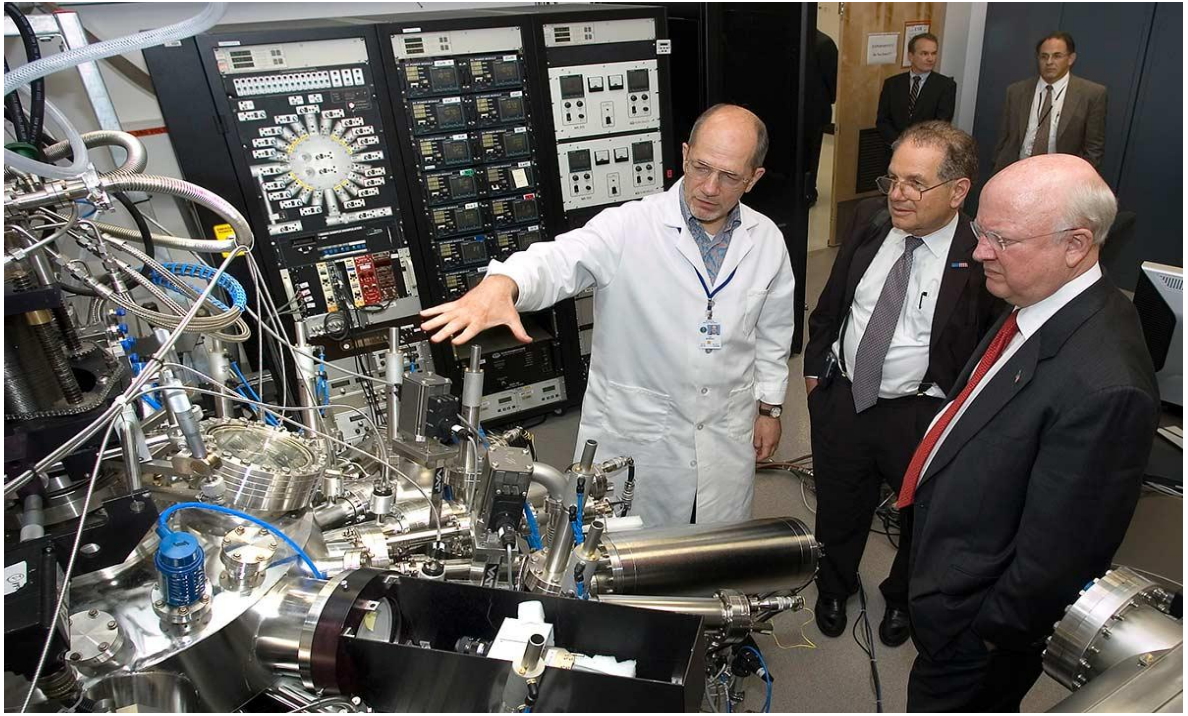
- Not json
- Not XML either
- Existing code can't be used

It’s quite simple...

- **Data types** (strings, numerals, arrays, maps)
- **References**
- **Methods**
  - Manipulate arrays/maps
  - External values (i.e. etcd)
- **Nested**
- **Recursive**

```
{
  Section_A: {
    #...
    Key_X: {
      dsl: “{max:{cref:Section_B, Key_Z}}”
    }
    #...
    Key_Y: {
      dsl: "{where:{etcd2folder:a/s/l}, 6}"
    }
  }
  Section_B: {
    #...
```

Oh boy



Source: https://www.bnl.gov/cmpmsd/mbe/

Regular Expressions

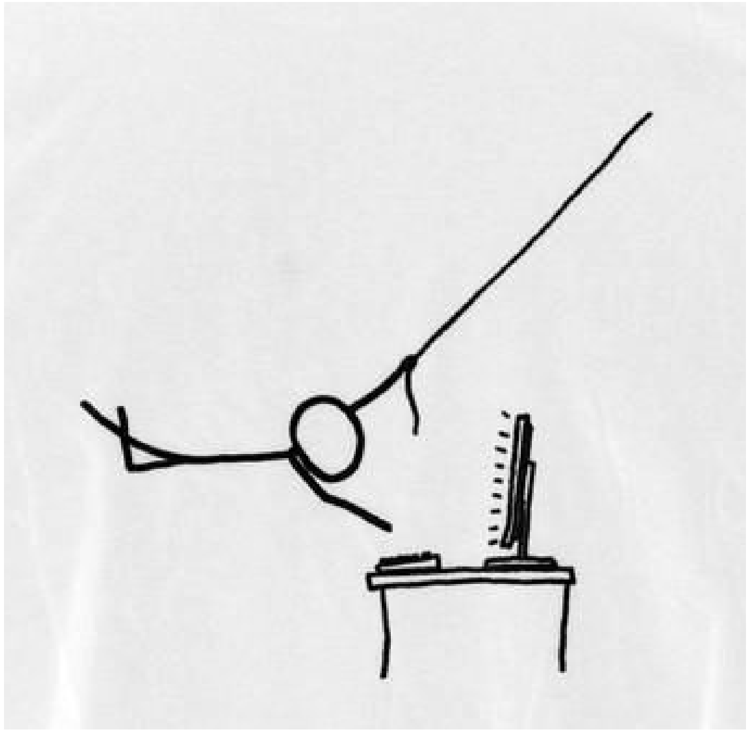
I know regular expressions

- Developer superpower
- Pattern matching
- Used for:

  Validation

  String Replacement

  “Parsing”



Source: https://xkcd.com/208/

(Simplified) INI file

```
[section]
key=value
key2=value2
[another_section]
foo=bar
```

(Simplified) Regular Expression

```
if re.match(‘\[(\w+)\]’, line):
    # <section stuff>
elif re.match(‘(\w+)=(\w+)’, line):
    # <key-value stuff>
```

```
[section]

key=value

key2=value2

[another_section]

foo=bar
```

Can I use it?

- Regular Languages
- From CS theory / Linguistics

    A language which can be **validated** in **O(1) space**

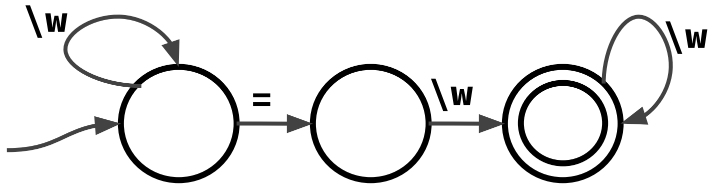
- Recognized by
    - Finite Automaton
    - Regular Expression

Regular or Not Regular?

INI key-value pairs

‘some_key=some_value’

“(\w+)=(\w+)”



INI key-value pairs where key and value match

‘some_key=some_key’

Not Regular

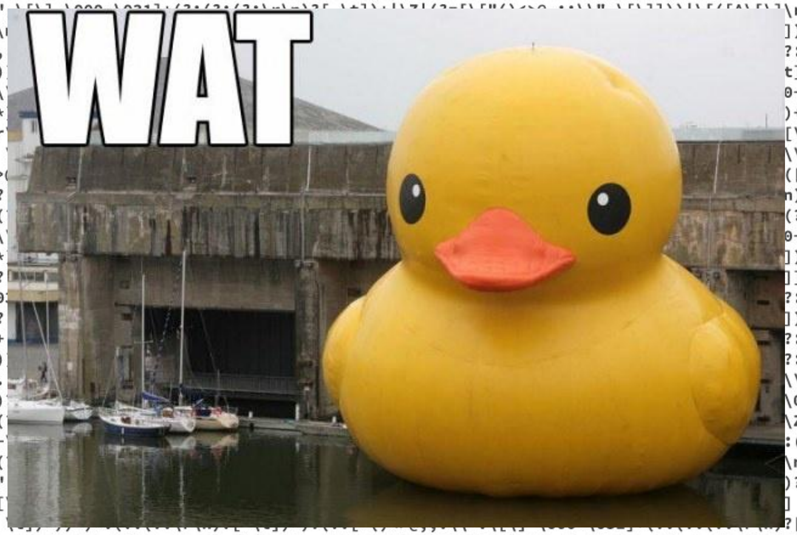
Theory Aside

```
>>> import re
>>> re.match(r'(\w+)=\1',
             'some_key=some_key')
<_sre.SRE_Match object at 0x7fb357fe25d0>
```

More **awesome sauce** can be found in [Matthew Barnett’s regex module]

Should I use RegEx?

```
(?:(?:\r\n)?[ \t])*(?:(?:(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)(?:\.(?:(?:
\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t])*(
?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\03
1]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*|(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[
\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)*\<(?:(?:\r\n)?[ \t])*(?:@(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([
^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)
?[ \t])*))*(?:,@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\"                                            \r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[
 \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\                                             ])*))*)*:(?:(?:\r\n)?[ \t])*)?(?:[^()<>@,;:\\".\[\]
 \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,                                             ?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+
(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]])                                             t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)
?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\                                             0-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\
".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*                                             )+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|
(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)*:(?:(?:\r                                             [\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\
n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[                                              \\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"
(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t])*(?:[^()<>                                             ([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:
\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?                                             n)?[ \t])*))*|(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(
?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(                                             (?:@(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)
?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\                                             0-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\
".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*                                             ])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\
.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?                                             ]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*)*
:(?:(?:\r\n)?[ \t])*)?(?:[^()<>@,;:\\".\[\] \000-\0                                             ?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\
n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?                                             ]))*"(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t])*(?:[
^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+                                             ?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+
(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]])                                             ?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t
])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.                                             \\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\
["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)                                             \031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".
\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(                                             \Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\
](?:(?:\r\n)?[ \t])*))*|(?:[^()<>@,;:\\".\[\] \000-                                             :(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)*\<(?:(?:\r\
n)?[ \t])*(?:@(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(                                             \n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\
".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["                                             )?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:
\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[                                             ] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@
,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[                                              ?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\
]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:
\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t
])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*\>(?:(?:\r\n)?[ \t])*))
*)?;\s*)
```



Source: http://blog.codinghorror.com/regex-use-vs-regex-abuse/

Should I use RegEx?

- The **iterative** coffee test
    - Make it readable: verbose (re.X), comments, named-groups
- Wrapper code
    - Common pattern: regex in loop
- Better alternatives?
    - Parsers

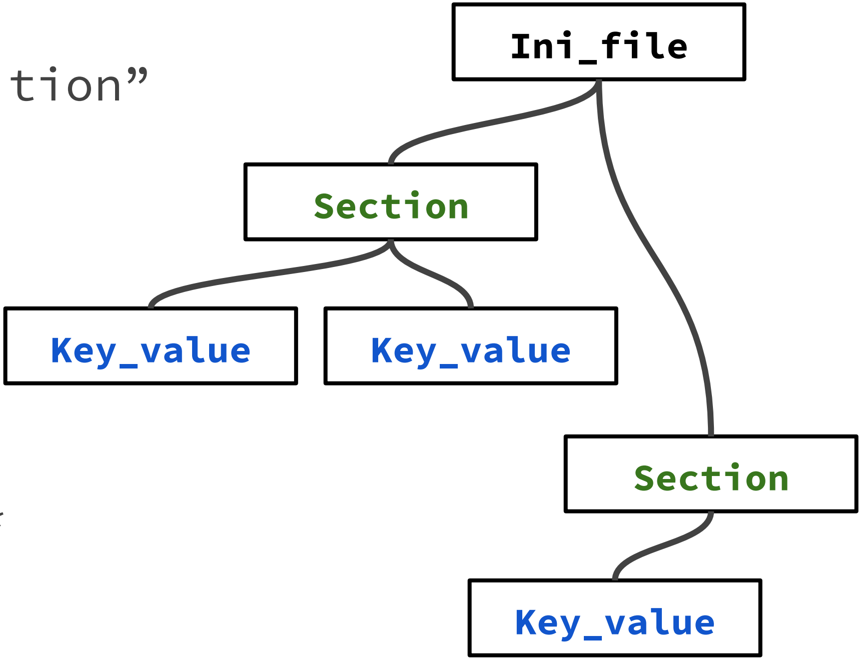
Source: http://broncrab.deviantart.com/art/Hulk-punches-Thor-308252233

Parsers

def parser(data, grammar): return tree

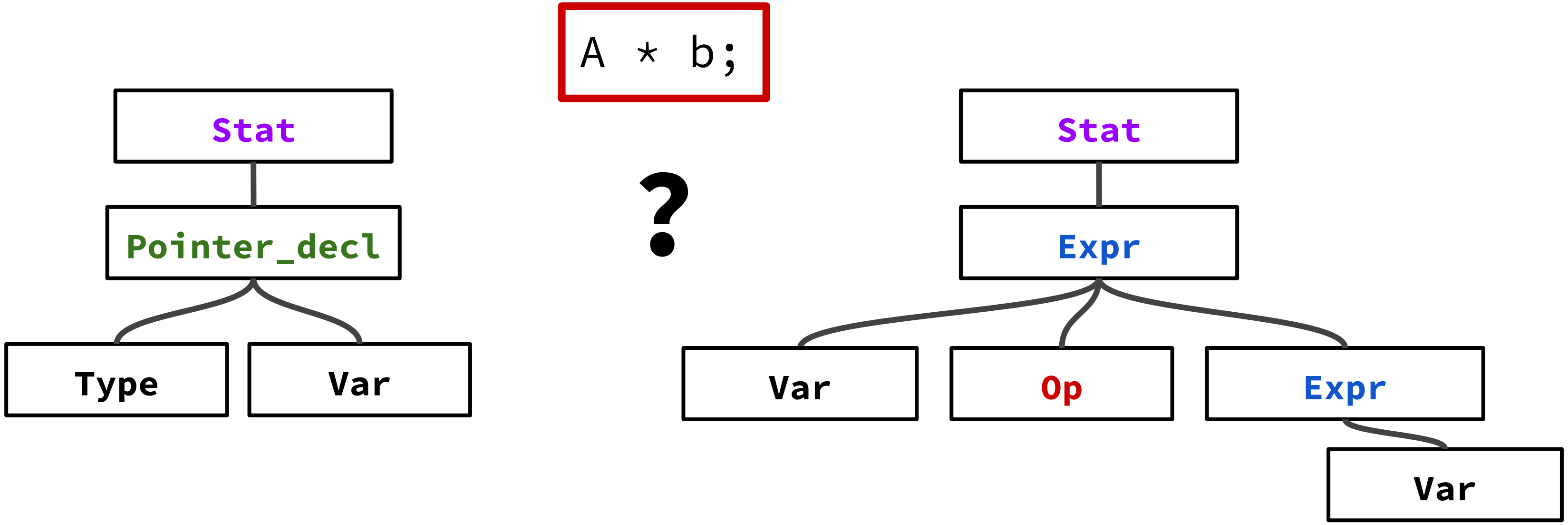
- Parsing: “Structural Decomposition”
- Grammar defines the structure
- Example:

```
Ini_file <- Section*
Section <- [\w+] \n Key_value*
Key_value <- \w+=\w+ \n
```

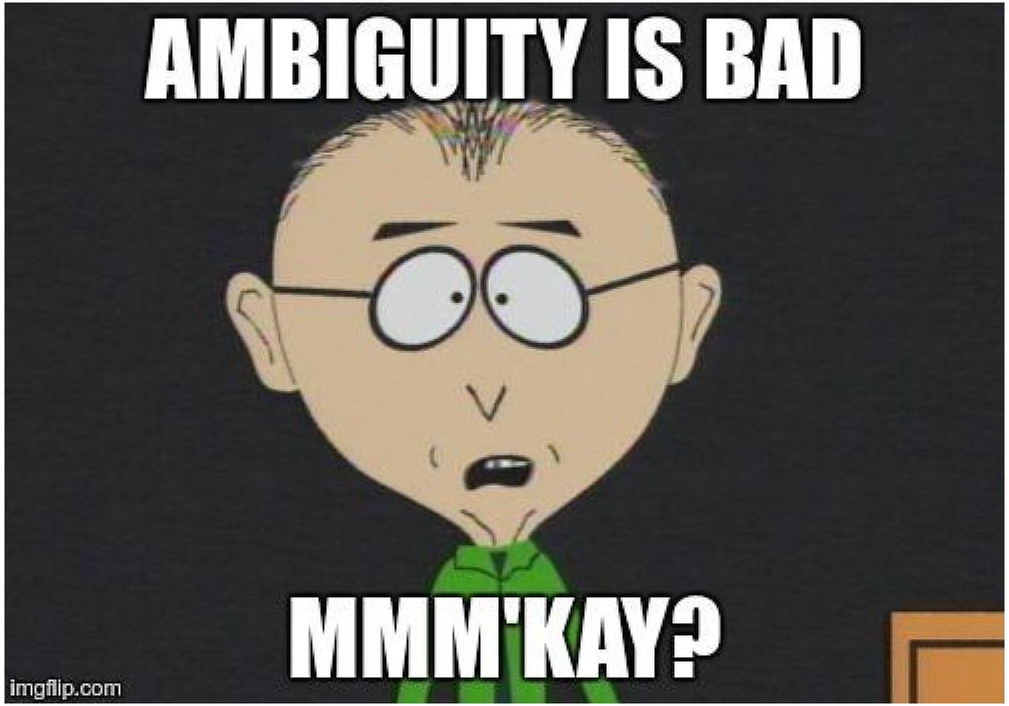


Grammar Ambiguity

When you have more than one way to parse



Grammar Ambiguity



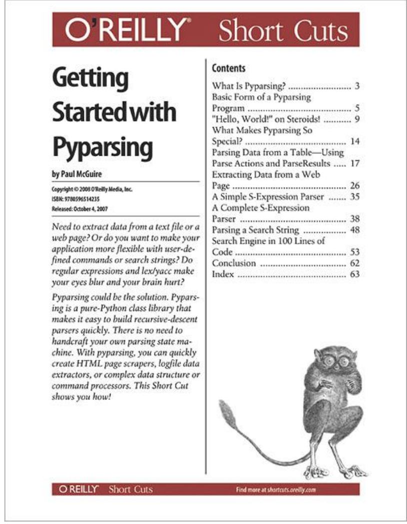
Choosing a parser

- Grammar Expressiveness
- QuickStart
- ~~Complexity~~
  - Time
  - Space

import pyparsing

———

```
lbrack = Literal("[").suppress()
rbrack = Literal("]").suppress()
equals = Literal("=").suppress()
semi   = Literal(";")
comment = semi + Optional( restOfLine )
nonrbrack = "".join( [ c for c in printables if c != "]" ] ) + " \t"
nonequals = "".join( [ c for c in printables if c != "=" ] ) + " \t"
sectionDef = lbrack + Word( nonrbrack ) + rbrack
keyDef = ~lbrack + Word( nonequals ) + equals + restOfLine
inibnf = Dict( ZeroOrMore( Group( sectionDef + Dict( ZeroOrMore( Group( keyDef
) ) ) ) ) )
iniFile = file(strng)
iniData = "".join( iniFile.readlines() )
bnf = inifile_BNF()
tokens = bnf.parseString( iniData )
```


O'REILLY Short Cuts

Getting Started with Pyparsing

by Paul McGuire

Copyright © 2008 O'Reilly Media, Inc.
ISBN: 9780596514235
Released: October 4, 2007

Need to extract data from a text file or a web page? Or do you want to make your application more flexible with user-defined commands or search strings? Do regular expressions and lex/yacc make your eyes blur and your brain hurt?

Pyparsing could be the solution. Pyparsing is a pure-Python class library that makes it easy to build recursive-descent parsers quickly. There is no need to handcraft your own parsing state machine. With pyparsing, you can quickly create HTML page scrapers, logfile data extractors, or complex data structure or command processors. This Short Cut shows you how!

Contents

O'REILLY Short Cuts Find more at shortcuts.oreilly.com

Source: https://pyparsing.wikispaces.com/Examples

import parsimonious

- PEG parser by Eric Rose
  - PEG == No Ambiguity
  - Designed to parse MediaWiki
- Parsing Horrible Things @ PyCon US 12
  - Including comparison to existing parsers
- Easy to use!

```
from parsimonious import Grammar

Grammar(my_rules).parse(my_data) # -> tree
```

Example: grammar

```
ini_grammar = parsimonious.Grammar(r"""
    file = section*
    section = "[" text "]" "\n" key_values
    key_values = key_value*
    key_value = text "=" text "\n"
    text = ~"[\w]*"
""")
```

Example: parser

```
with open('config.ini') as text_file:
    tree = ini_grammar.parse(text_file.read())
```

Example: output

```
[another_section]

foo=bar
```

```
---

<Node called "section" matching "...">
        <Node matching "[">
        <RegexNode called "text" matching "another_section">
        <Node matching "]">
        #...
            <Node called "key_value" matching "...">
                <RegexNode called "text" matching "foo">
                <Node matching "=">
                <RegexNode called "text" matching "bar">
                #...
```

Climbing trees

```
class ININodeVisitor(NodeVisitor):
    def generic_visit(self, node, visited_children):
        pass  # For unspecified visits, return None
    def visit_text(self, node, visited_children):
        return node.text  # text rule
    def visit_key_value(self, node, visited_children):
        return tuple([e for e in visited_children if e is not None])
```

Climbing trees

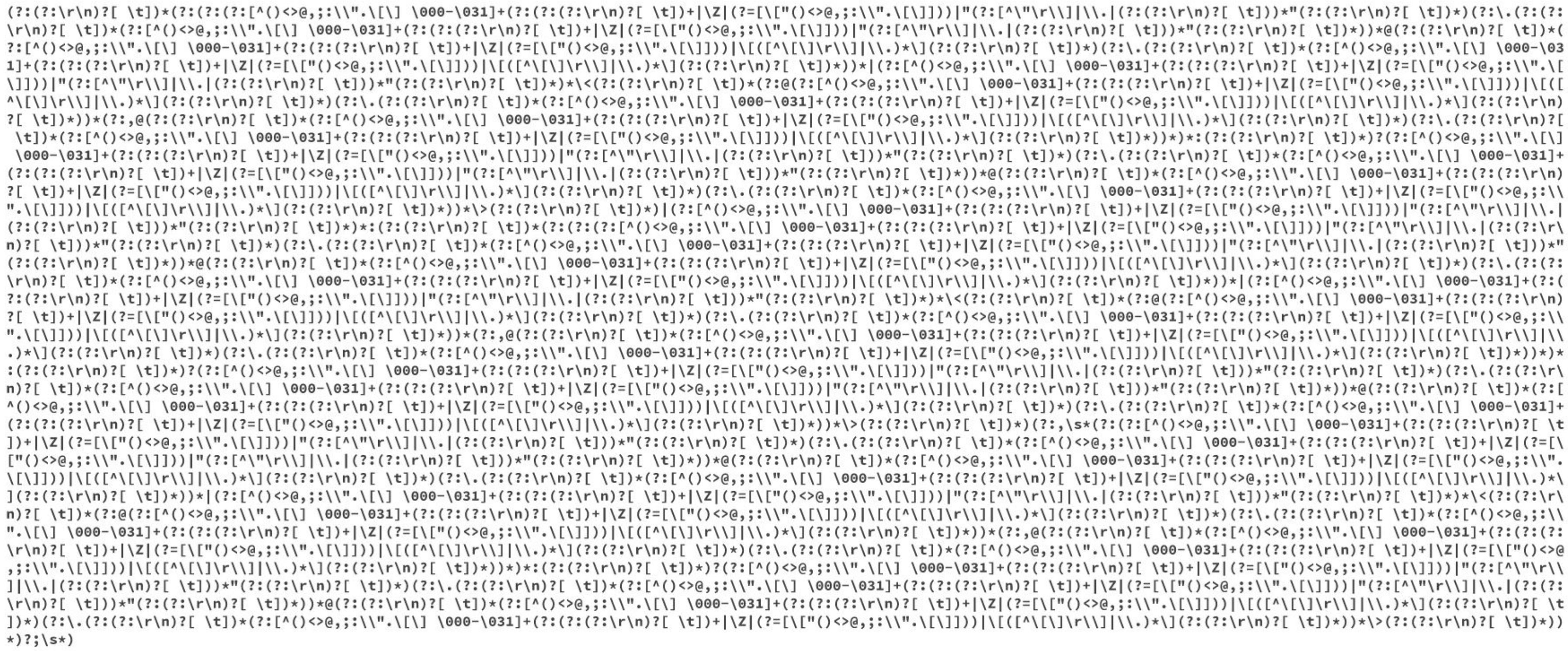
```
#...
   def visit_key_values(self, node, visited_children):
       return dict(e for e in visited_children if e is not None)
   #...
nv = ININodeVisitor()
print nv.visit(tree)  # {‘another_section’: {‘foo’: ‘bar’}}
```

Common pitfalls

```
A = B / "foo"
B = C
C = A
```

- Avoiding circular definitions
- Parsing exceptions can be vague
- NodeVisitor documentation is lacking
  - "For now, have a look at its docstrings for more detail"
  - ast.NodeVisitor() doesn't add much

Still better than this



```
(?:(?:\r\n)?[ \t])*(?:(?:(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)(?:\.(?:(?:
\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t])*(
?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\03
1]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*|(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[
\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)*\<(?:(?:\r\n)?[ \t])*(?:@(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([
^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)
?[ \t])*))*(?:,@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[
 \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*)*:(?:(?:\r\n)?[ \t])*)?(?:[^()<>@,;:\\".\[\]
 \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+
(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)
?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\
".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*\>(?:(?:\r\n)?[ \t])*)|(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|
(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)*:(?:(?:\r\n)?[ \t])*(?:(?:(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\
n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"
(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:
\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*|(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(
?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)*\<(?:(?:\r\n)?[ \t])*(?:@(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)
?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\
".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*(?:,@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\
.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*)*
:(?:(?:\r\n)?[ \t])*)?(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\
n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t])*(?:[
^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+
(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*\>(?:(?:\r\n)?[ \t])*)(?:,\s*(?:(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t
])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\
["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".
\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\
](?:(?:\r\n)?[ \t])*))*|(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)*\<(?:(?:\r\
n)?[ \t])*(?:@(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\
".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*(?:,@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:
\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@
,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*)*:(?:(?:\r\n)?[ \t])*)?(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\
]|\\.|(?:(?:\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|"(?:[^\"\r\\]|\\.|(?:(?:
\r\n)?[ \t]))*"(?:(?:\r\n)?[ \t])*))*@(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t
])*)(?:\.(?:(?:\r\n)?[ \t])*(?:[^()<>@,;:\\".\[\] \000-\031]+(?:(?:(?:\r\n)?[ \t])+|\Z|(?=[\["()<>@,;:\\".\[\]]))|\[([^\[\]\r\\]|\\.)*\](?:(?:\r\n)?[ \t])*))*\>(?:(?:\r\n)?[ \t])*))
*)?;\s*)
```

Source: http://blog.codinghorror.com/regex-use-vs-regex-abuse/

Summary

- Regular Expressions are far more
- Don’t fear the Parser
    - Fear leads to .* suffering
- Now you have two hammers!

Source: https://retcon-punch.com/2013/07/25/thor-god-of-thunder-10/

Thank You!

Nati Cohen (@nocoot)

References

- Eric Rose
    - erikrose/parsimonious
    - Parsing Horrible Things with Python (PyCon US 2012) [Video] [Slides]
    - Python parser comparison (w/ Peter Potrowl, 8/2011)
- Ford, Bryan. "Parsing expression grammars: a recognition-based syntactic foundation." ACM SIGPLAN Notices. Vol. 39. No. 1. ACM, 2004. [paper]

References

- PEG.js a simple parser generator for JavaScript

NOTE: import regex

```
>>> json_pattern = r'''
...   (?(DEFINE)
...      (?<number>   -? (?= [1-9]|0(?!\d) ) \d+ (\.\d+)? ([eE] [+-]? \d+)? )
...      (?<boolean>   true | false | null )
...      (?<string>    " ([^"\\\\]* | \\\\ ["\\\\bfnrt\/] | \\\\ u [0-9a-f]{4} )* " )
...      (?<array>     \[  (?:  (?&json)  (?: , (?&json)  )*  )?  \s* \] )
...      (?<pair>      \s* (?&string) \s* : (?&json)  )
...      (?<object>    \{  (?:  (?&pair)  (?: , (?&pair)  )*  )?  \s* \} )
...      (?<json>   \s* (?: (?&number) | (?&boolean) | (?&string) | (?&array) | (?&object) ) \s* )
...   )
...   ^ (?&json) $
... '''
# Read data ...
>>> regex.match(json_pattern, data, regex.V1 | regex.X)
<regex.Match object; ... >
```

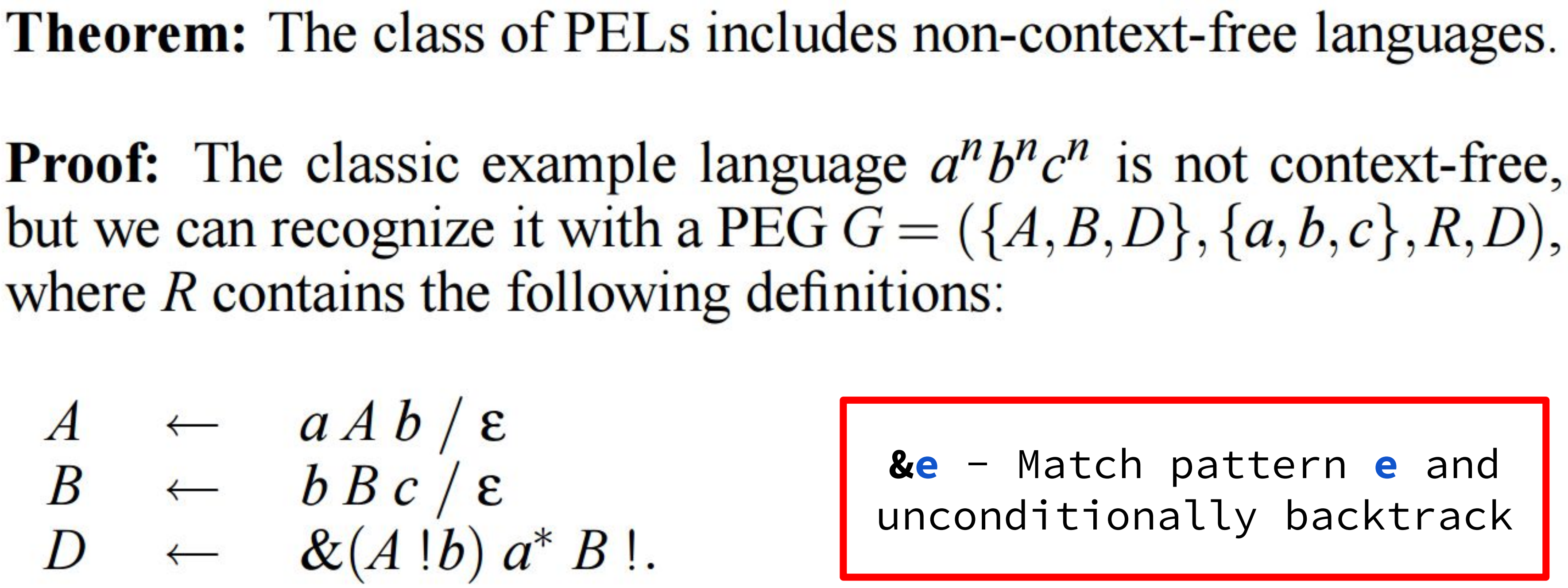
Source: http://stackoverflow.com/questions/2583472/regex-to-validate-json

NOTE: Parsers are not always

```
>>> import urlparse
>>> urlparse.urlparse('http://Hi :: PyCon!.il').netloc
'Hi :: PyCon!.il'
```

See Django’s URLValidator

NOTE: PEG vs CFG



Theorem: The class of PELs includes non-context-free languages.

Proof: The classic example language $a^n b^n c^n$ is not context-free, but we can recognize it with a PEG $G = (\{A, B, D\}, \{a, b, c\}, R, D)$, where R contains the following definitions:

$$\begin{array}{lcl} A & \leftarrow & a\,A\,b\ /\ \varepsilon \\ B & \leftarrow & b\,B\,c\ /\ \varepsilon \\ D & \leftarrow & \&(A\,!b)\ a^*\ B\ !. \end{array}$$

&e - Match pattern **e** and unconditionally backtrack