



# Computational Structures in Data Science

---

## Lecture #13: Regular Expressions





**UC Berkeley EECS 246  
Adj. Assistant Prof.  
Dr. Donald Redwood**

Facebook... \*sigh\*

April 15th, 2018

<http://info.eecs.berkeley.edu/cs246>

## Speaking of Facebook...



<https://www.youtube.com/watch?v=bghuioP8hs0>

<http://www.teachingprivacy.org>

©2013 ISTE

LEARNING BYTES LLC

## Another Grammar

---


$$S \rightarrow aSb$$
$$S \rightarrow \epsilon$$

- $S$  non-terminal
- $a, b$  terminal
- Epsilon: empty!

Question: What are valid words in this grammar?

## Recap: Language Structures (Python)

- **Variables and literals**
  - with some internal representation, e.g. integers, floats, booleans, strings.
  - In Python: implicit data types!
- **Operations on variable and literals of a type**
  - e.g.: `x = 2 + 3` `y = 3.14` `z = "hello"`
- **Operations are valid well-defined sets of operations on variables and literals that produce a value of a type.**
  - `a = 4 + 3`



## Computational Concepts Toolbox

- Data type: values, literals, operations,
- Expressions, Call expression
- Variables
- Assignment Statement
- Sequences: tuple, list
- Dictionaries
- Data structures
- Tuple assignment
- Function Definition
- Statement
- Conditional Statement
- Iteration: list comp, for, while
- Lambda function expr.
- Higher Order Functions  
as Values, Argc, Results
- Higher order function patterns
  - Map, Filter, Reduce
  - Function factories
- Recursion
  - Linear, Tail, Tree
- Abstract Data Types
- Mutation
- Iterators and Generators
- Object Oriented Programming, Classes
- Exceptions
- Declarative Programming
- Regular Expressions

© 2014 KTH ROYAL INSTITUTE OF TECHNOLOGY

# On Languages...

- **Human (Natural Language)**
  - Developed by (social) evolution
  - Used to communicate between humans
  - Change "brain state" of recipient of communication
- **Mathematics**
  - Formal language developed out of philosophy
  - Syntax (structure) and semantics (meaning) well defined
  - Used to communicate scientific results between humans more rigorously
- **Programming Languages**
  - Formalized grammar allows automatic translation
  - Syntax and semantics unambiguous but limited
  - Used to communicate between humans and computer

02/11/18 UCSD/CMU Spring 1.3

## Recap: while statement – iteration control


---

- Repeat a block of statements until a predicate expression is satisfied

```

<Initialization statements>
while <predicate expression>:
    <body statements>

<rest of the program>
                    
```



4/10/2019
UCON CS464 Sp19/L12

**Grammar Hierarchy**

The diagram illustrates the hierarchy of languages and their corresponding automata:

- Outermost Level:** grammars (generators) and languages (left) and automata (acceptors) (right).
- Second Level:** recursively - enumerable language (left) and Turing machine (right).
- Third Level:** context-sensitive language (left) and linear bounded automaton (right).
- Fourth Level:** context-free language (left) and push-down automaton (right).
- Fifth Level:** regular language (left) and finite-state automaton (right).

The diagram shows that each level of automata is contained within the level of languages it accepts, and each level of languages is contained within the level of grammars that generate them.

the automata hierarchy

# Syntax, Grammar, Semantics

- **Syntax (programming language):**
  - Set of rules that defines the combinations of symbols that are considered to be a correctly structured document or fragment in a programming language.
- **Grammar:**
  - Formalism (language) that defines the syntax of a programming language.
- **Semantics:**
  - Consequence (meaning) attached to a sequence of symbols.

[illegible]

## Grammar Hierarchy

Language	Grammar	Automation
Regular	$A = a$	Finite state machine
Context-free	$A = \{ \}$	Non-deterministic pushdown automaton
Context-sensitive	$aA \rightarrow a\beta$	Liner bounded non-deterministic Turing machine
Recursively enumerable	$\alpha \rightarrow \beta$ no restrictions	Turing machine

- Programming Languages usually context-free
- Many filter tools for data usually regular languages (i.e. form regular expressions)

## Regular Expressions

- Easy to parse
- Semantics typically:
  - Find <string>
  - Find <string1> Replace with <string2>
- Widely available in:
  - Python
    - Unix command line (bash, flex, sed)
    - Various editors (emacs, vi)
    - Most data science tools
- Important mechanism for 'cleaning' or 'extracting' data

## Regular Expressions: Math

- Invited by S. Kleene
- Most important operator: \* (say 'star' or Kleene star)
- s\* means: 's' n-times (for varying n)

## Regular Expressions: Python

- re package
- Regular expressions are compiled into object
- Then various methods are available:

```
>>> import re
>>> p = re.compile("r")
>>> p
re.compile("r")

>>> print(p.match("r"))
None

>>> m = p.match("sasses")
>>> m
<_sre.SRE_Match object; span=(0, 5),
match="sasses">
```

## More on REs in Python

```
>>> re.split("W+", "Words, words, words.")
['Words', 'words', 'words']
>>> re.split("W+", "Words, words, words.", flags=re.IGNORECASE)
['Words', ' ', 'words', ' ', 'words', ' ', '']
>>> re.split("W+", "Words, words, words.", 1)
['Words', 'words', ' ', 'words', ' ', '']
>>> re.split("[a-z]", "abc3567", flags=re.IGNORECASE)
['', 'a', 'b', 'c']

>>> re.split("(W?+", "words, words.", 1)
['', ' ', 'words', ' ', ' ', ' ', '']

>>> re.split("w+", "foo")
['foo']

>>> re.split("(?u)", "foolishbar")
['foolishbar']
```

[illegible]

## Summary: RegEx

- Programming Languages are very formally defined using grammars
- Types of grammars need different complexity
- Regular expressions very simple and effective tool for data filtering
- More in the lab!

• Next Lecture: Information and Bits