Regular Expressions

- A pattern language for matching input and extracting subtext.
- Usable within any programming language
- Abbreviations: regex, regexp
- Makes life easier!

Regexes and Langauges

- Perl5, natively supported
- Java
- Python
- C++
  - http://code.google.com/p/re2/
- *NIX
  - Grep
  - Sed
  - ed

Basic Pattern Elements

- Literal
  - Letters and numbers
  - Special characters preceded by \
  - Matches itself
- Character Class
  - [abc] matches any of the letters a, b, or c
  - [a-gmx-z] matches any letters in the range a-g, or x-z. Also matches the letter m.
  - . matches anything except the \n character.

Basic Pattern Elements

- Quantifier
  - {1,4} matches one to four of the preceding element.
  - {2,} matches 2 or more of the preceding element.
  - * a shorthand for {0,}
  - + a shorthand for {1,}
  - ? a shorthand for {0,1}

Basic Pattern Elements

- Groups
  - (apple|pear) matches "apple" or "pear"
  - (ant|bat|) matches "ant", "bat", or ""
  - Groups are numbered, based on the order of the opening parenthesis within the expression, and can be used to extract submatches.

Examples in Python

```
import re
p = re.compile("(apple|pear)")
n = 0
while True:
    # The n identifies where to start searching
    m = p.search("apple,pear,apple,pear,pear",n)
    if m:
        n = m.end() # find the position of the match end
        print m.group() # print the matched text
    else:
        break
```

Examples in Python

```
import re
p = re.compile("a*b")
n = 0
while True:
    m = p.search("aabaaababb",n)
    if m:
        n = m.end()
        print m.group()
    else:
        break
```

Examples in Python

```
import re
p = re.compile("\\([0-9]{3}\\)[0-9]{3}-[0-9]{4}")
n = 0
while True:
    m = p.search("11987(201)730-83736642",n)
    if m:
        n = m.end()
        print m.group()
    else:
        break
```

Examples in Python

```
import re
p = re.compile("\\(([0-9]{3})\\)([0-9]{3})-([0-9]{4})")
n = 0
while True:
    m = p.search("11987(201)730-83736642",n)
    if m:
        n = m.end()
        g = m.groups() # extract submatches
        print 'match:',m.group()
        for i in range(len(g)):
            print i,g[i]
    else:
        break
```

Examples in Python

```
import re
p = re.compile("(sna(kes*|ail)|(b(at|ird)))")
n = 0
while True:
    m = p.search("birdbatsnakesssnail",n)
    if m:
        n = m.end()
        print m.group()
    else:
        break
```

Other Pattern elements

- \d matches [0-9]
- \D matches anything but [0-9]
- [^0-9] matches anything but [0-9]
- \s matches a whitespace character, e.g. space, tab, new line
- \w matches a C-language identifier character, e.g. letter, number, or underscore, [a-zA-Z0-9_]
- \W matches anything but \w

Other Pattern Elements

- ^ matches the start of a string
- $ matches the end of a string
- \b matches a word boundary, i.e. A boundary between a C identifier character and a non C-identifier character, or the beginning or end of a string
- \B matchs if not a word boundary
- (?i) turn on ignore case mode
- (?m) multiline mode

Examples in Python

```
import re
p = re.compile("^start")
n = 0
while True:
    m = p.search("start\nstart\nstart\nstart",n)
    if m:
        n = m.end()
        print m.group()
    else:
        break
```

Examples in Python

```
import re
p = re.compile("(?m)^start")
n = 0
while True:
    m = p.search("start\nstart\nstart\nstart",n)
    if m:
        n = m.end()
        print m.group()
    else:
        break
```

Examples in Python

```
import re
p = re.compile("\\bcat")
n = 0
while True:
    m = p.search("cat cat bobcat",n)
    if m:
        n = m.end()
        print m.group()
    else:
        break
```

Examples in Python

```
import re
p = re.compile("\\Bcat")
n = 0
while True:
    m = p.search("cat cat bobcat",n)
    if m:
        n = m.end()
        print m.group()
    else:
        break
```

Examples in C++

```
#include <iostream>
#include <string>
#include "re2/re2.h"
#include "re2/stringpiece.h"
#include <assert.h>
int main() {
    int i; std::string s; RE2 re("((?i)apple)");
    re2::StringPiece txt("appleAppleAPPLE");
    std::cout << "Start Matching" << std::endl;
    while(RE2::Consume(&txt,re,&s)) {
        std::cout << "Match: " << s << std::endl;
    }
    std::cout << "End Matching" << std::endl;
    return 0;
}
```

Examples in Java

```
import java.util.regex.*;

public class Phone {
    public static void main(String[] args) {
        Pattern p = Pattern.compile(
                "(\\([0-9]{3}\\)|)(\\d{3})-(\\d{4})");
        Matcher m = p.matcher(
                "my num is 123-4567 or (301)123-4567");
        while(m.find()) {
            System.out.println("match=<"+m.group()+">");
            for(int n=1;n<=m.groupCount();n++)
                System.out.println("   "+n+"="+m.group(n));
        }
    }
}
```

Take Questions

- Address
- Mathematical Expression
- Proper name
- Email address
- Social Security Number