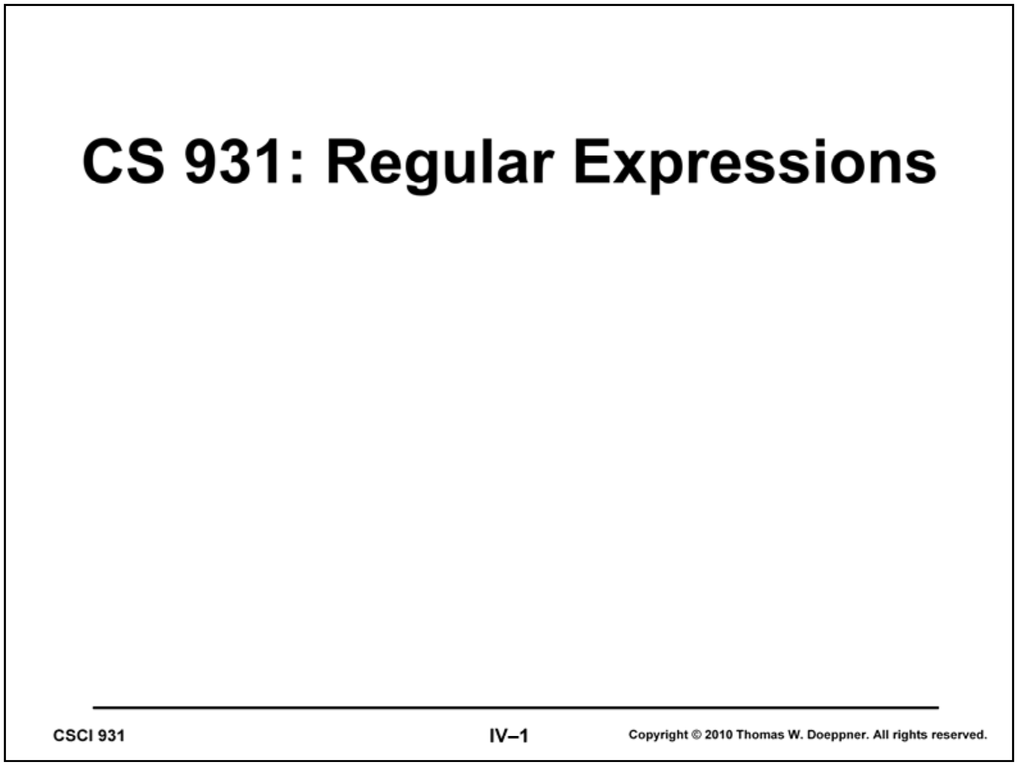
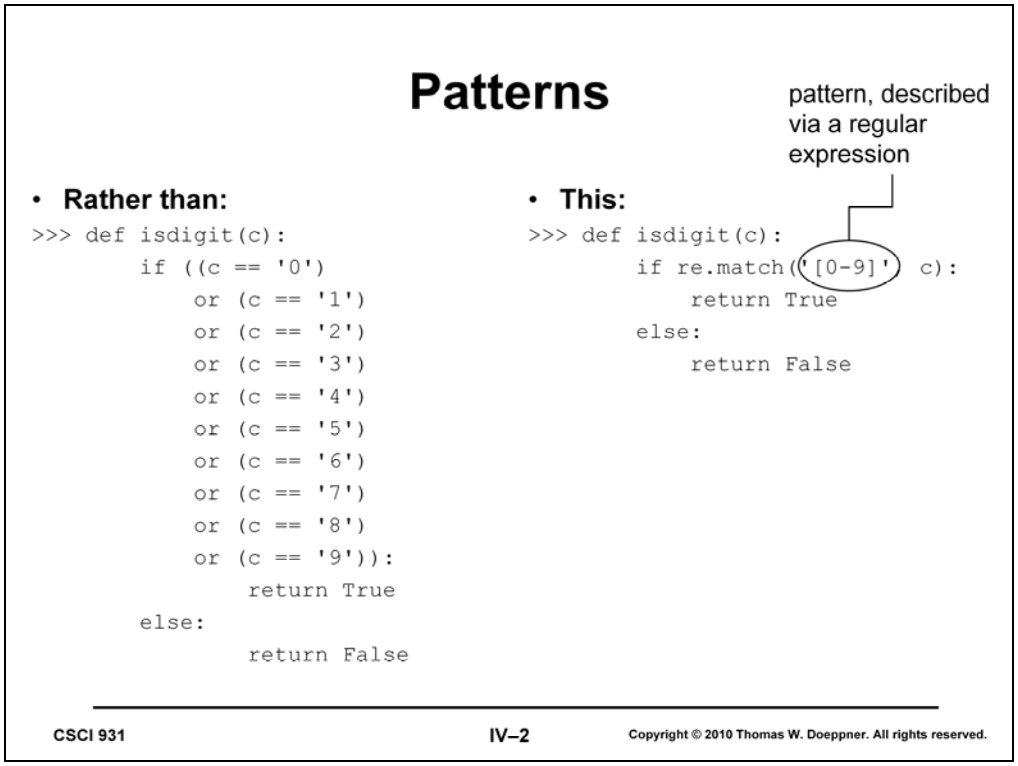


CS 931: Regular Expressions

More than you ever wanted to know about regular expressions can be found at http://docs.python.org/library/re.html.



Patterns

pattern, described via a regular expression

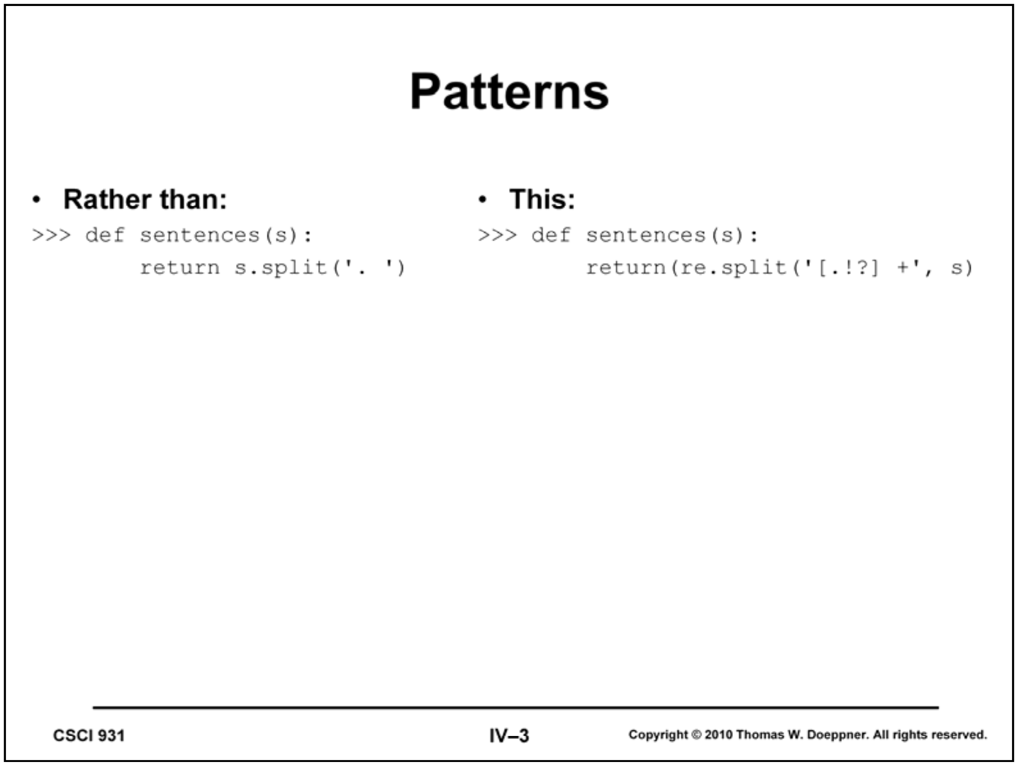
- **Rather than:**

```
>>> def isdigit(c):
        if ((c == '0')
            or (c == '1')
            or (c == '2')
            or (c == '3')
            or (c == '4')
            or (c == '5')
            or (c == '6')
            or (c == '7')
            or (c == '8')
            or (c == '9')):
                return True
        else:
                return False
```

- **This:**

```
>>> def isdigit(c):
        if re.match('[0-9]'  c):
            return True
        else:
            return False
```

The pattern in the right column ('[0-9]') matches any of the characters 0 through 9.



Patterns

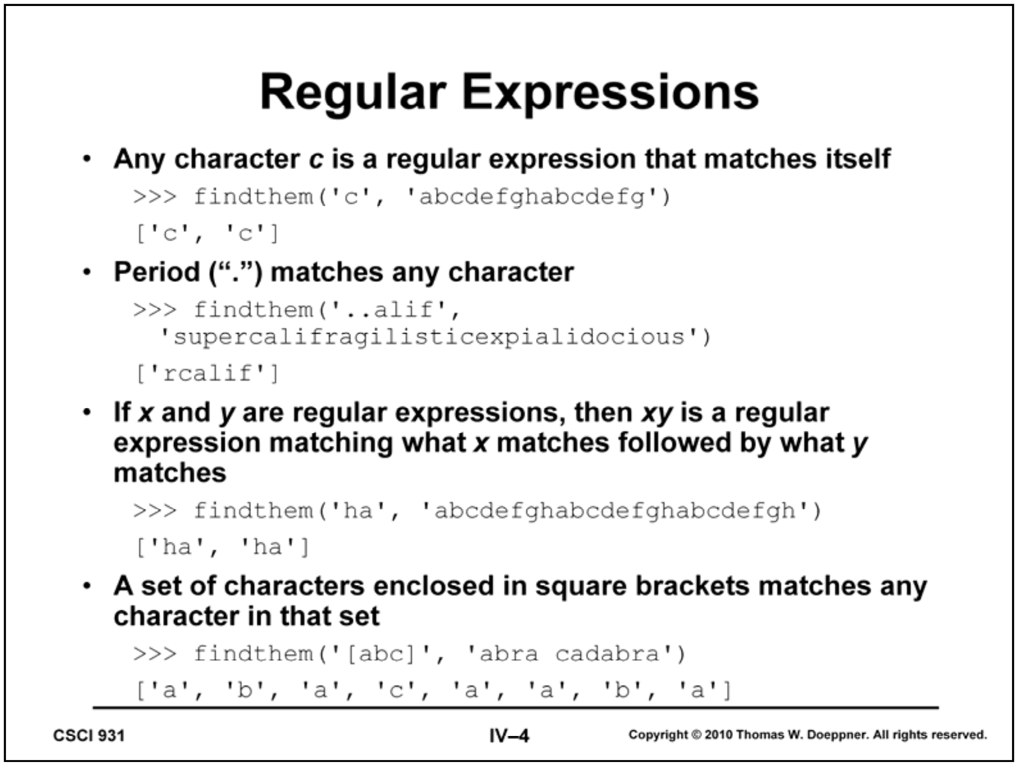
- **Rather than:**

```
>>> def sentences(s):
        return s.split('. ')
```

- **This:**

```
>>> def sentences(s):
        return(re.split('[.!?] +', s)
```

The pattern in the right column matches any of period, exclamation point, and question mark, followed by one or more spaces. (Note that if the text contains line breaks ('\n' characters), then the pattern that delineates sentences is a bit more complicated.)



Regular Expressions

- **Any character *c* is a regular expression that matches itself**

```
>>> findthem('c', 'abcdefghabcdefg')
['c', 'c']
```

- **Period (“.”) matches any character**

```
>>> findthem('..alif',
  'supercalifragilisticexpialidocious')
['rcalif']
```

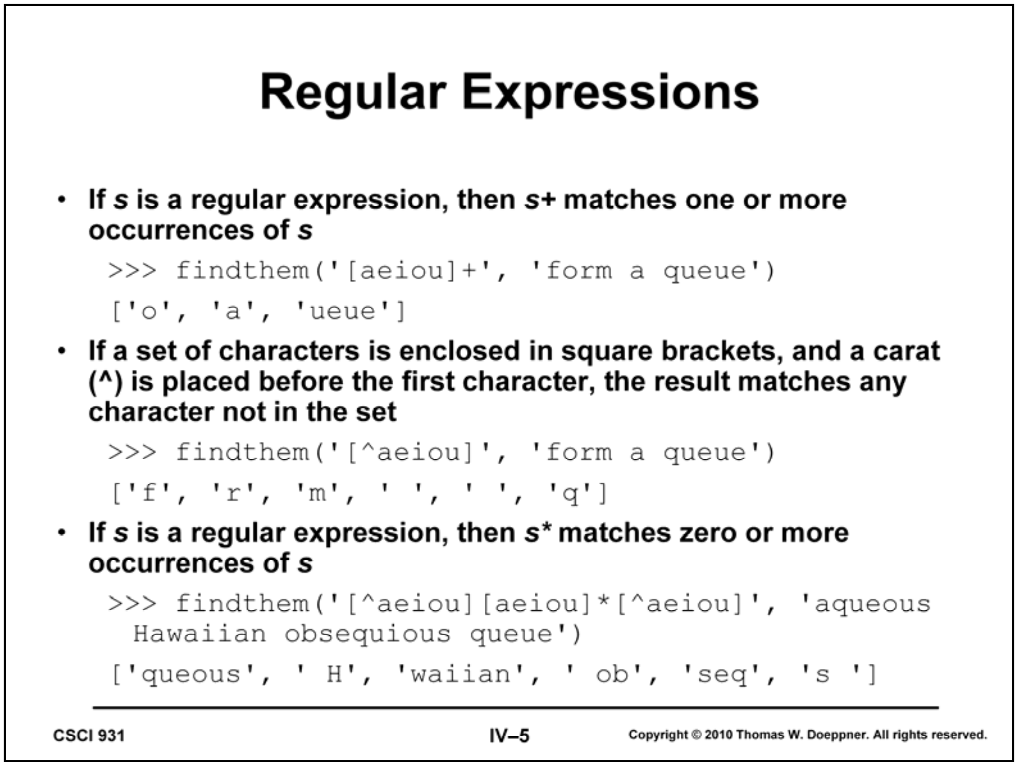
- **If *x* and *y* are regular expressions, then *xy* is a regular expression matching what *x* matches followed by what *y* matches**

```
>>> findthem('ha', 'abcdefghabcdefghabcdefgh')
['ha', 'ha']
```

- **A set of characters enclosed in square brackets matches any character in that set**

```
>>> findthem('[abc]', 'abra cadabra')
['a', 'b', 'a', 'c', 'a', 'a', 'b', 'a']
```

findthem is a Python function we will give you that produces a list of everything matched by the first argument (a regular expression) in the second argument (a string).



Regular Expressions

- If *s* is a regular expression, then *s*+ matches one or more occurrences of *s*

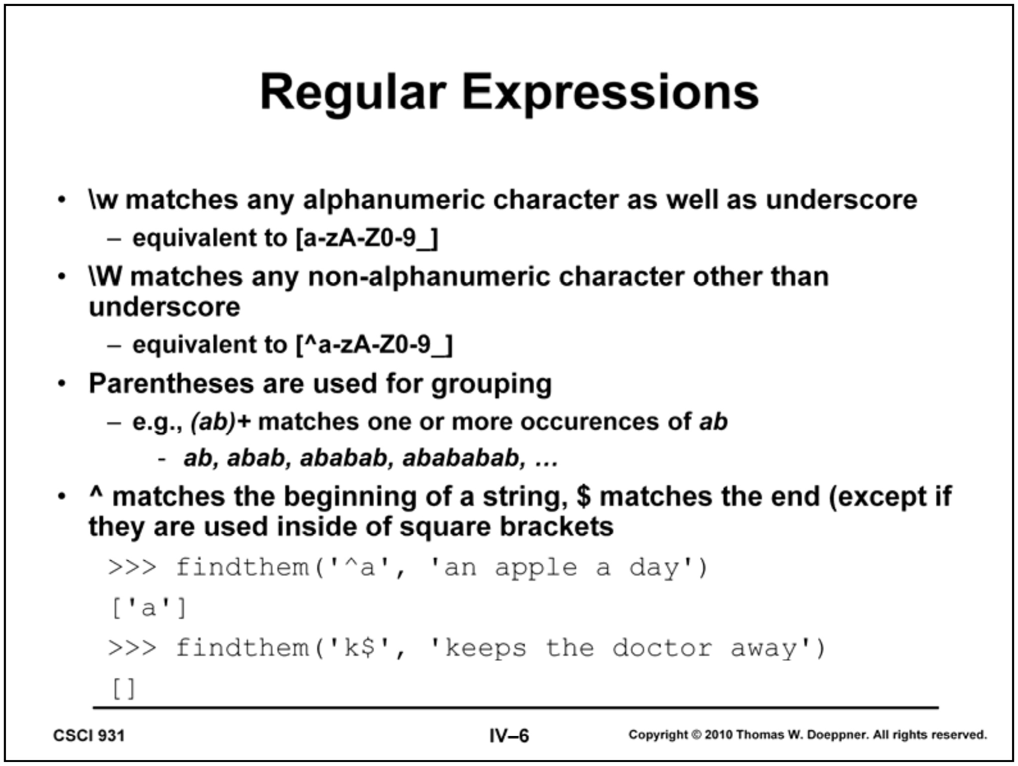
```
>>> findthem('[aeiou]+', 'form a queue')
['o', 'a', 'ueue']
```

- If a set of characters is enclosed in square brackets, and a carat (^) is placed before the first character, the result matches any character not in the set

```
>>> findthem('[^aeiou]', 'form a queue')
['f', 'r', 'm', ' ', ' ', 'q']
```

- If *s* is a regular expression, then *s** matches zero or more occurrences of *s*

```
>>> findthem('[^aeiou][aeiou]*[^aeiou]', 'aqueous
  Hawaiian obsequious queue')
['queous', ' H', 'waiian', ' ob', 'seq', 's ']
```



Regular Expressions

- \w matches any alphanumeric character as well as underscore
  - equivalent to [a-zA-Z0-9_]
- \W matches any non-alphanumeric character other than underscore
  - equivalent to [^a-zA-Z0-9_]
- Parentheses are used for grouping
  - e.g., *(ab)+* matches one or more occurences of *ab*
    - *ab, abab, ababab, abababab, …*
- ^ matches the beginning of a string, $ matches the end (except if they are used inside of square brackets

```
>>> findthem('^a', 'an apple a day')
['a']
>>> findthem('k$', 'keeps the doctor away')
[]
```

Regular Expressions

- **If *x* and *y* are regular expressions, then *x*|*y* matches either *x* or *y***

```
>>> findthem('(cat)|(dog)', 'nice doggie')
['dog']
>>> findthem("(\w['\w]*\w)|(\w+)", "doesn't this
  work?")
["doesn't", 'this', 'work']
```