

Week 2 - Dynamic Functions

```
# examples with the regular expressions module

import re

s = "mississippi"
match = re.search('miss', s)
print match.start(), match.end(), match.group()

# special characters
# .      match a single character
# *      match 0 or more of the previous character
# +      match 1 or more of the previous character
# [abc]  matches a or b or c
# [^ab]  matches anything but a or b
# [a-z]  matches any character a-z
# (ab)+  matches 1 or more "ab"
# ^      matches the start of the line
# $      matches the end of the line

match = re.search('is*', s)
print match.start(), match.end(), match.group()

match = re.search("(is*)+", s)
print match.start(), match.end(), match.group()

match = re.search(r"(i[sp]*)+", s)
print match.start(), match.end(), match.group()

s = "mississippi"
print re.sub("[aeiou]", "o", s)
print re.sub("is+", "ipp", s)
```

The following code examples are from Mark Pilgrim's *Dive Into Python*. We'll be looking at code that generates the plural form of English words.

1 Stage 1

```
import re

def plural(noun):
    if re.search('[sxz]$', noun):
        return re.sub('$', 'es', noun)
    elif re.search('[^aeioudgkprt]h$', noun):
        return re.sub('$', 'es', noun)
    elif re.search('[^aeiou]y$', noun):
        return re.sub('y$', 'ies', noun)
    else:
        return noun + 's'
```

2 Stage 2

```
import re

def match_sxz(noun):
    return re.search('[sxz]$', noun)

def apply_sxz(noun):
    return re.sub('$', 'es', noun)

def match_h(noun):
    return re.search('[^aeiou]h$', noun)

def apply_h(noun):
    return re.sub('$', 'es', noun)

def match_y(noun):
    return re.search('[^aeiou]y$', noun)

def apply_y(noun):
    return re.sub('y$', 'ies', noun)

def match_default(noun):
    return 1

def apply_default(noun):
    return noun + 's'

rules = ((match_sxz, apply_sxz),
         (match_h, apply_h),
         (match_y, apply_y),
         (match_default, apply_default)
        )

def plural(noun):
    for matchesRule, applyRule in rules:
        if matchesRule(noun):
            return applyRule(noun)
```

3 Stage 3

```
import re

rules = \
(
  (
    lambda word: re.search('[sxz]$', word),
    lambda word: re.sub('$', 'es', word)
  ),
  (
    lambda word: re.search('[^aeiou]h$', word),
    lambda word: re.sub('$', 'es', word)
  ),
  (
    lambda word: re.search('[^aeiou]y$', word),
    lambda word: re.sub('y$', 'ies', word)
  ),
  (
    lambda word: re.search('$', word),
    lambda word: re.sub('$', 's', word)
  )
)

def plural(noun):
  for matchesRule, applyRule in rules:
    if matchesRule(noun):
      return applyRule(noun)
```

4 Stage 4

```
import re

def buildMatchAndApplyFunctions((pattern, search, replace)):
    matchFunction = lambda word: re.search(pattern, word)
    applyFunction = lambda word: re.sub(search, replace, word)
    return (matchFunction, applyFunction)

patterns = \
(
    ('[sxz]$', '$', 'es'),
    ('^[^aeiou]h$', '$', 'es'),
    ('(qu|[^aeiou])y$', 'y$', 'ies'),
    ('$', '$', 's')
)
rules = map(buildMatchAndApplyFunctions, patterns)

def plural(noun):
    for matchesRule, applyRule in rules:
        if matchesRule(noun):
            return applyRule(noun)
```

5 Stage 5

rules.en

```
[sxz]$           $           es
[^aeiou]h$      $           es
[^aeiou]y$      y$         ies
$               $           s
```

plural5.py

```
import re
import string
def buildRule((pattern, search, replace)):
    return lambda word: re.search(pattern, word) and re.sub(search, replace, word)

def plural(noun, language='en'):
    lines = file('rules.%s' % language).readlines()
    patterns = map(string.split, lines)
    rules = map(buildRule, patterns)
    for rule in rules:
        result = rule(noun)
        if result: return result
```