

Counting Words & Pattern Matching

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

1 Dictionaries
free books as .txt files
dictionaries in Python
sorting dictionary items

2 Pattern Matching
using the `re` module
matching strings with `match()`

3 Regular Expressions
common regular expression symbols
groups of regular expressions

MCS 275 Lecture 7
Programming Tools and File Management
Jan Verschelde, 27 January 2010

Counting Words Pattern Matching

Dictionaries

free books as .txt files

dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

1 Dictionaries
free books as .txt files
dictionaries in Python
sorting dictionary items

2 Pattern Matching
using the `re` module
matching strings with `match()`

3 Regular Expressions
common regular expression symbols
groups of regular expressions

War of the Worlds

Dictionaries

free books as .txt files

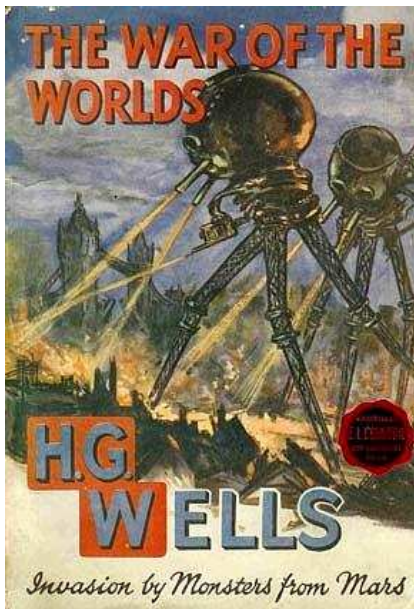
dictionaries in Python
sorting dictionary items

Pattern Matching

using the re module
matching strings with match()

Regular Expressions

common regular expression symbols
groups of regular expressions



1898 novel

text in public domain

www.gutenberg.org

downloaded .txt file

for this course:

word frequencies

“war of words”

Dictionaries

free books as .txt files

dictionaries in Python
sorting dictionary items

Pattern Matching

using the re module
matching strings with match()

Regular Expressions

common regular expression symbols
groups of regular expressions

War of Words

frequencies of words

Processing a text file:

- 1 How many different words?
- 2 We compute the frequency for each word.
- 3 For each frequency, we list of the words.
- 4 We list all words used 100 times or more.

To us, anything separated by spaces is a word.

Results of the Analysis

Dictionaries

free books as .txt files

dictionaries in Python
sorting dictionary items

Pattern Matching

using the re module
matching strings with match()

Regular Expressions

common regular expression symbols
groups of regular expressions

Total words counted: 66,491

Number of different words: 13,003

Words used more than 100 times:

```
(4076, ['the'])
```

```
...
```

```
(119, ['there'])
```

```
(116, ['people', 'And'])
```

```
(114, ['an'])
```

```
(113, ['Martians'])
```

```
(111, ['saw', 'through'])
```

```
...
```

Scanning Files

Dictionaries

free books as .txt files

dictionaries in Python
sorting dictionary items

Pattern Matching

using the re module
matching strings with match()

Regular Expressions

common regular expression symbols
groups of regular expressions

```
book = "war_of_the_worlds.txt"
```

```
def word_count(name):  
    """  
    Opens the file with name and counts the  
    number of words. Anything that is separated  
    by spaces is considered a word.  
    """  
    f = open(name, 'r')  
    cnt = 0  
    while True:  
        s = f.readline()  
        if s == "": break  
        L = s.split(' ')  
        cnt = cnt + len(L)  
    f.close()  
    return cnt
```

Scanning Files

Dictionaries

free books as .txt files

dictionaries in Python
sorting dictionary items

Pattern Matching

using the re module
matching strings with match()

Regular Expressions

common regular expression symbols
groups of regular expressions

```
book = "war_of_the_worlds.txt"
```

```
def word_count(name):  
    """  
    Opens the file with name and counts the  
    number of words. Anything that is separated  
    by spaces is considered a word.  
    """  
    f = open(name, 'r')  
    cnt = 0  
    while True:  
        s = f.readline()  
        if s == "": break  
        L = s.split(' ')  
        cnt = cnt + len(L)  
    f.close()  
    return cnt
```

Counting Words

Pattern Matching

Dictionaries

free books as .txt files

dictionaries in Python

sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

1 Dictionaries

free books as .txt files

dictionaries in Python

sorting dictionary items

2 Pattern Matching

using the `re` module

matching strings with `match()`

3 Regular Expressions

common regular expression symbols

groups of regular expressions

Dictionaries in Python

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

A dictionary in Python is ...

- a set of `key:value` pairs
any type goes for `value`
but `key` must belong to an ordered type
- a hash table where order of elements allows for fast access.

Our application: frequency table

- type of `key`: `str`
- type of `value`: `int`

An example of a `key:value` pair:

`'the': 4076`

Dictionaries in Python

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

A dictionary in Python is ...

- a set of `key:value` pairs
any type goes for `value`
but `key` must belong to an ordered type
- a hash table where order of elements allows for fast access.

Our application: frequency table

- type of `key`: `str`
- type of `value`: `int`

An example of a `key:value` pair:

`'the': 4076`

Using Dictionaries

Dictionaries

free books as .txt files

dictionaries in Python
sorting dictionary items

Pattern Matching

using the re module
matching strings with match()

Regular Expressions

common regular expression symbols
groups of regular expressions

```
>>> D = {}
>>> D['s'] = 1
>>> D.has_key('s')
True
>>> D.has_key('t')
False
>>> D['t'] = 2

>>> D.items()
[('s', 1), ('t', 2)]
>>> D.values()
[1, 2]
>>> D.keys()
['s', 't']
```

Using Dictionaries

Dictionaries

free books as .txt files

dictionaries in Python
sorting dictionary items

Pattern Matching

using the re module
matching strings with match()

Regular Expressions

common regular expression symbols
groups of regular expressions

```
>>> D = {}
>>> D['s'] = 1
>>> D.has_key('s')
True
>>> D.has_key('t')
False
>>> D['t'] = 2

>>> D.items()
[('s', 1), ('t', 2)]
>>> D.values()
[1, 2]
>>> D.keys()
['s', 't']
```

Useful Constructions

Dictionaries

free books as .txt files

dictionaries in Python

sorting dictionary items

Pattern Matching

using the re module
matching strings with match()

Regular Expressions

common regular expression symbols
groups of regular expressions

Python	what it means
<code>D = { }</code>	initialization
<code>D[<key>] = <value></code> <code>D[<key>]</code>	add a key:value pair selection of value, given key
<code>D.has_key(<key>)</code>	returns True or False depending on whether D[<key>] exists
<code>D.items()</code>	list of tuples (key, value)
<code>D.keys()</code>	return list of all keys
<code>D.values()</code>	return list of all values

Useful Constructions

Dictionaries

free books as .txt files

dictionaries in Python

sorting dictionary items

Pattern Matching

using the re module
matching strings with
match()

Regular Expressions

common regular
expression symbols
groups of regular
expressions

Python	what it means
<code>D = { }</code>	initialization
<code>D[<key>] = <value></code>	add a key:value pair
<code>D[<key>]</code>	selection of value, given key
<code>D.has_key(<key>)</code>	returns True or False depending on whether <code>D[<key>]</code> exists
<code>D.items()</code>	list of tuples (key, value)
<code>D.keys()</code>	return list of all keys
<code>D.values()</code>	return list of all values

Useful Constructions

Dictionaries

free books as .txt files

dictionaries in Python

sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

Python	what it means
<code>D = { }</code>	initialization
<code>D[<key>] = <value></code>	add a <code>key:value</code> pair
<code>D[<key>]</code>	selection of value, given key
<code>D.has_key(<key>)</code>	returns True or False depending on whether <code>D[<key>]</code> exists
<code>D.items()</code>	list of tuples (key, value)
<code>D.keys()</code>	return list of all keys
<code>D.values()</code>	return list of all values

a Dictionary of Frequencies

Dictionaries

free books as .txt files

dictionaries in Python

sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

```
def word_frequencies(name):
    """
    Returns a dictionary with the frequencies
    of the words occurring on file with name.
    """
    f = open(name, 'r'); D = {}
    while True:
        s = f.readline()
        if s == "": break
        L = s.split(' ')
        for e in L:
            if D.has_key(e):
                D[e] = D[e] + 1
            else:
                D[e] = 1
    f.close()
    return D
```

a Dictionary of Frequencies

Dictionaries

free books as .txt files

dictionaries in Python

sorting dictionary items

Pattern Matching

using the re module
matching strings with match()

Regular Expressions

common regular expression symbols
groups of regular expressions

```
def word_frequencies(name):  
    """  
    Returns a dictionary with the frequencies  
    of the words occurring on file with name.  
    """  
    f = open(name, 'r'); D = {}  
    while True:  
        s = f.readline()  
        if s == "": break  
        L = s.split(' ')  
        for e in L:  
            if D.has_key(e):  
                D[e] = D[e] + 1  
            else:  
                D[e] = 1  
    f.close()  
    return D
```

Composite Values

Dictionaries

free books as .txt files

dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

With `D = word_frequencies('book')`
we see how many times each word occurs,
for example: `D['the']` returns 4076.

But we want to know the most frequent words,
that is: we want to query `D` on the values.

We revert the dictionary:

- 1 the keys are the frequency counts,
- 2 because words occur more than once,
the values are lists.

For example: `F[295]` will be `['for', 'from']`,
if `F` is the reverted `D`.

Composite Values

Dictionaries

free books as .txt files

dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

With `D = word_frequencies('book')`
we see how many times each word occurs,
for example: `D['the']` returns 4076.

But we want to know the most frequent words,
that is: we want to query `D` on the values.

We revert the dictionary:

- 1 the keys are the frequency counts,
- 2 because words occur more than once,
the values are lists.

For example: `F[295]` will be `['for', 'from']`,
if `F` is the reverted `D`.

Frequencies as Keys

Dictionaries

free books as .txt files

dictionaries in Python
sorting dictionary items

Pattern Matching

using the re module
matching strings with match()

Regular Expressions

common regular expression symbols
groups of regular expressions

```
def frequencies_of_words(D):  
    """  
    Reverts the keys and values of the  
    given dictionary D.  
    Because several words may occur with  
    the same frequency, the values are  
    lists of words.  
    """  
    F = {}  
    for k in D:  
        if F.has_key(D[k]):  
            F[D[k]].append(k)  
        else:  
            F[D[k]] = [k]  
    return F
```

Frequencies as Keys

Dictionaries

free books as .txt files

dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

```
def frequencies_of_words(D):  
    """  
    Reverts the keys and values of the  
    given dictionary D.  
    Because several words may occur with  
    the same frequency, the values are  
    lists of words.  
    """  
    F = {}  
    for k in D:  
        if F.has_key(D[k]):  
            F[D[k]].append(k)  
        else:  
            F[D[k]] = [k]  
    return F
```

Counting Words

Pattern Matching

Dictionaries

free books as .txt files

dictionaries in Python

sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

1 Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

2 Pattern Matching

using the `re` module
matching strings with `match()`

3 Regular Expressions

common regular expression symbols
groups of regular expressions

Sorting Dictionary Items

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

Recall our goal: words used more than 100 times.

The `items()` method on any dictionary returns a list of tuples:

```
>>> L = D.items()
>>> L
[('s', 1), ('t', 2)]
```

To sort on a key, from high to low:

```
>>> L.sort(key=lambda i: i[1],reverse=True)
>>> L
[('t', 2), ('s', 1)]
```

Sorting Dictionary Items

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the re module
matching strings with match()

Regular Expressions

common regular expression symbols
groups of regular expressions

Recall our goal: words used more than 100 times.

The `items()` method on any dictionary returns a list of tuples:

```
>>> L = D.items()
>>> L
[('s', 1), ('t', 2)]
```

To sort on a key, from high to low:

```
>>> L.sort(key=lambda i: i[1],reverse=True)
>>> L
[('t', 2), ('s', 1)]
```

the main program

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the re module
matching strings with match()

Regular Expressions

common regular expression symbols
groups of regular expressions

```
def main():
    """
    Analysis of words in a book.
    """
    cnt = word_count(book)
    print "words counted : ", cnt
    D = word_frequencies(book)
    print "number of different words :", len(D)
    F = frequencies_of_words(D)
    L = F.items()
    L.sort(key=lambda e: e[0],reverse=True)
    print "words used more than 100 times :"
    for e in L:
        if e[0] < 100: break
    print e
```

the main program

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the re module
matching strings with match()

Regular Expressions

common regular expression symbols
groups of regular expressions

```
def main():
    """
    Analysis of words in a book.
    """
    cnt = word_count(book)
    print "words counted : ", cnt
    D = word_frequencies(book)
    print "number of different words :", len(D)
    F = frequencies_of_words(D)
    L = F.items()
    L.sort(key=lambda e: e[0],reverse=True)
    print "words used more than 100 times :"
    for e in L:
        if e[0] < 100: break
    print e
```

the main program

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the re module
matching strings with match()

Regular Expressions

common regular expression symbols
groups of regular expressions

```
def main():
    """
    Analysis of words in a book.
    """
    cnt = word_count(book)
    print "words counted : ", cnt
    D = word_frequencies(book)
    print "number of different words :", len(D)
    F = frequencies_of_words(D)
    L = F.items()
    L.sort(key=lambda e: e[0],reverse=True)
    print "words used more than 100 times :"
    for e in L:
        if e[0] < 100: break
    print e
```

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

Counting Words

Pattern Matching

1 Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

2 Pattern Matching

using the `re` module
matching strings with `match()`

3 Regular Expressions

common regular expression symbols
groups of regular expressions

Regular Expressions

using the `re` module

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

Manipulating text and strings is an important task:

- parse to ensure entered data is correct,
- search through confidential data: use program.

Suppose `answer` contains the answers to a yes or no question.

Acceptable yes answers:

- 1 y or yes
- 2 Y or Yes

Testing all these cases is tedious.

Support for regular expressions:

```
>>> import re
```

`re` is a standard library module.

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

Regular Expressions

using the `re` module

Manipulating text and strings is an important task:

- parse to ensure entered data is correct,
- search through confidential data: use program.

Suppose `answer` contains the answers to a yes or no question.

Acceptable yes answers:

- 1 y or yes
- 2 Y or Yes

Testing all these cases is tedious.

Support for regular expressions:

```
>>> import re
```

`re` is a standard library module.

Regular Expressions

using the `re` module

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

Manipulating text and strings is an important task:

- parse to ensure entered data is correct,
- search through confidential data: use program.

Suppose `answer` contains the answers to a yes or no question.

Acceptable *yes* answers:

- 1 `y` or `yes`
- 2 `Y` or `Yes`

Testing all these cases is tedious.

Support for regular expressions:

```
>>> import re
```

`re` is a standard library module.

Regular Expressions

using the `re` module

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

Manipulating text and strings is an important task:

- parse to ensure entered data is correct,
- search through confidential data: use program.

Suppose `answer` contains the answers to a yes or no question.

Acceptable yes answers:

- 1 `y` or `yes`
- 2 `Y` or `Yes`

Testing all these cases is tedious.

Support for regular expressions:

```
>>> import re
```

`re` is a standard library module.

Regular Expressions

using the `re` module

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

Manipulating text and strings is an important task:

- parse to ensure entered data is correct,
- search through confidential data: use program.

Suppose `answer` contains the answers to a yes or no question.

Acceptable yes answers:

- 1 `y` or `yes`
- 2 `Y` or `Yes`

Testing all these cases is tedious.

Support for regular expressions:

```
>>> import re
```

`re` is a standard library module.

Matching short and long Answers

using `re` module

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

```
>>> import re
>>> short = 'y'; long = 'Yes'
>>> re.match('y',short) != None
True
>>> re.match('y',long) != None
False
>>> re.match('y|Y',long) != None
True
>>> re.match('y|Y',long)
<_sre.SRE_Match object at 0x5cb10>
>>> re.match('y|Y',long).group()
'y'
>>> re.match('y|Y',short).group()
'y'
```

Matching short and long Answers

using `re` module

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

```
>>> import re
>>> short = 'y'; long = 'Yes'
>>> re.match('y',short) != None
True
>>> re.match('y',long) != None
False
>>> re.match('y|Y',long) != None
True
>>> re.match('y|Y',long)
<_sre.SRE_Match object at 0x5cb10>
>>> re.match('y|Y',long).group()
'y'
>>> re.match('y|Y',short).group()
'y'
```

Matching short and long Answers

using `re` module

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

```
>>> import re
>>> short = 'y'; long = 'Yes'
>>> re.match('y',short) != None
True
>>> re.match('y',long) != None
False
>>> re.match('y|Y',long) != None
True
>>> re.match('y|Y',long)
<_sre.SRE_Match object at 0x5cb10>
>>> re.match('y|Y',long).group()
'y'
>>> re.match('y|Y',short).group()
'y'
```

Matching short and long Answers

using `re` module

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

```
>>> import re
>>> short = 'y'; long = 'Yes'
>>> re.match('y',short) != None
True
>>> re.match('y',long) != None
False
>>> re.match('y|Y',long) != None
True
>>> re.match('y|Y',long)
<_sre.SRE_Match object at 0x5cb10>
>>> re.match('y|Y',long).group()
'y'
>>> re.match('y|Y',short).group()
'y'
```

Matching short and long Answers

using `re` module

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

```
>>> import re
>>> short = 'y'; long = 'Yes'
>>> re.match('y',short) != None
True
>>> re.match('y',long) != None
False
>>> re.match('y|Y',long) != None
True
>>> re.match('y|Y',long)
<_sre.SRE_Match object at 0x5cb10>
>>> re.match('y|Y',long).group()
'y'
>>> re.match('y|Y',short).group()
'y'
```

Matching short and long Answers

using `re` module

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

```
>>> import re
>>> short = 'y'; long = 'Yes'
>>> re.match('y',short) != None
True
>>> re.match('y',long) != None
False
>>> re.match('y|Y',long) != None
True
>>> re.match('y|Y',long)
<_sre.SRE_Match object at 0x5cb10>
>>> re.match('y|Y',long).group()
'y'
>>> re.match('y|Y',short).group()
'y'
```

Matching short and long Answers

using `re` module

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

```
>>> import re
>>> short = 'y'; long = 'Yes'
>>> re.match('y',short) != None
True
>>> re.match('y',long) != None
False
>>> re.match('y|Y',long) != None
True
>>> re.match('y|Y',long)
<_sre.SRE_Match object at 0x5cb10>
>>> re.match('y|Y',long).group()
'y'
>>> re.match('y|Y',short).group()
'y'
```

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

Counting Words Pattern Matching

1 Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

2 Pattern Matching

using the `re` module
matching strings with `match()`

3 Regular Expressions

common regular expression symbols
groups of regular expressions

Matching Strings with `match()`

creating match objects

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

The function `match()` in the `re` module:

```
>>> re.match( < pattern > , < string > )
```

If the `string` does not match the `pattern`, then `None` is returned.

If the `string` matches the `pattern`, then a match object is returned.

```
>>> re.match('he', 'hello')  
<_sre.SRE_Match object at 0x5cb10>  
>>> re.match('hi', 'hello') == None  
True
```

Matching Strings with `match()`

creating match objects

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

The function `match()` in the `re` module:

```
>>> re.match( < pattern > , < string > )
```

If the string does not match the pattern, then `None` is returned.

If the string matches the pattern, then a match object is returned.

```
>>> re.match('he', 'hello')  
<_sre.SRE_Match object at 0x5cb10>  
>>> re.match('hi', 'hello') == None  
True
```

Matching Strings with `match()`

creating match objects

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

The function `match()` in the `re` module:

```
>>> re.match( < pattern > , < string > )
```

If the string does not match the pattern, then `None` is returned.

If the string matches the pattern, then a match object is returned.

```
>>> re.match('he', 'hello')  
<_sre.SRE_Match object at 0x5cb10>  
>>> re.match('hi', 'hello') == None  
True
```

Matching Strings with `match()`

creating match objects

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

The function `match()` in the `re` module:

```
>>> re.match( < pattern > , < string > )
```

If the string does not match the pattern, then `None` is returned.

If the string matches the pattern, then a match object is returned.

```
>>> re.match('he', 'hello')  
<_sre.SRE_Match object at 0x5cb10>  
>>> re.match('hi', 'hello') == None  
True
```

Matching Strings with `match()`

creating match objects

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

The function `match()` in the `re` module:

```
>>> re.match( < pattern > , < string > )
```

If the string does not match the pattern, then `None` is returned.

If the string matches the pattern, then a match object is returned.

```
>>> re.match('he', 'hello')  
<_sre.SRE_Match object at 0x5cb10>  
>>> re.match('hi', 'hello') == None  
True
```

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

The `group()` Method

using a match object

What can we do with the match object?

```
>>> re.match('he', 'hello')
<_sre.SRE_Match object at 0x5cb10>
>>> _.group()
'he'
```

After a successful match, `group()` returns that part of the pattern that matches the string.

The `group()` Method

using a match object

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

What can we do with the match object?

```
>>> re.match('he', 'hello')
<_sre.SRE_Match object at 0x5cb10>
>>> _.group()
'he'
```

After a successful match, `group()` returns that part of the pattern that matches the string.

Searching versus Matching

the methods `search()` and `match()`

Dictionaries

free books as .txt files

dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

The match only works from the start:

```
>>> re.match('ell', 'hello') == None
True
```

Looking for the first occurrence of the pattern in the string, with `search()`:

```
>>> re.search('ell', 'hello')
<_sre.SRE_Match object at 0x5cb10>
>>> _.group()
'ell'
```

Searching versus Matching

the methods `search()` and `match()`

Dictionaries

free books as .txt files

dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

The `match` only works from the start:

```
>>> re.match('ell', 'hello') == None
True
```

Looking for the first occurrence of the pattern in the string, with `search()`:

```
>>> re.search('ell', 'hello')
<_sre.SRE_Match object at 0x5cb10>
>>> _.group()
'ell'
```

Searching versus Matching

the methods `search()` and `match()`

Dictionaries

free books as .txt files

dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

The `match` only works from the start:

```
>>> re.match('ell', 'hello') == None
True
```

Looking for the first occurrence of the pattern in the string, with `search()`:

```
>>> re.search('ell', 'hello')
<_sre.SRE_Match object at 0x5cb10>
>>> _.group()
'ell'
```

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

Counting Words Pattern Matching

1 Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

2 Pattern Matching

using the `re` module
matching strings with `match()`

3 Regular Expressions

common regular expression symbols
groups of regular expressions

Regular Expressions

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

pattern	strings matched
<code>literal</code>	strings starting with <code>literal</code>
<code>re1 re2</code>	strings starting with <code>re1</code> or <code>re2</code>

```
>>> from time import ctime
>>> now = ctime()
>>> now
'Fri Apr 25 07:32:07 2008'
>>> p = ...
'\w{3}\s\w{3}\s\d{2}\s\d{2}:\d{2}:\d{2}\s\d{4}'
>>> re.match(p,now) != None
True
```

pattern	strings matched
<code>\w</code>	any alphanumeric character, same as <code>[A-Za-z]</code>
<code>\d</code>	any decimal digit, same as <code>[0-9]</code>
<code>\s</code>	any whitespace character
<code>re{n}</code>	n occurrences of <code>re</code>

Regular Expressions

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

pattern	strings matched
<code>literal</code>	strings starting with <code>literal</code>
<code>re1 re2</code>	strings starting with <code>re1</code> or <code>re2</code>

```
>>> from time import ctime
```

```
>>> now = ctime()
```

```
>>> now
```

```
'Fri Apr 25 07:32:07 2008'
```

```
>>> p = ...
```

```
'\w{3}\s\w{3}\s\d{2}\s\d{2}:\d{2}:\d{2}\s\d{4}'
```

```
>>> re.match(p,now) != None
```

```
True
```

pattern	strings matched
<code>\w</code>	any alphanumeric character, same as <code>[A-Za-z]</code>
<code>\d</code>	any decimal digit, same as <code>[0-9]</code>
<code>\s</code>	any whitespace character
<code>re{n}</code>	n occurrences of <code>re</code>

Regular Expressions

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

pattern	strings matched
<code>literal</code>	strings starting with <code>literal</code>
<code>re1 re2</code>	strings starting with <code>re1</code> or <code>re2</code>

```
>>> from time import ctime
>>> now = ctime()
>>> now
'Fri Apr 25 07:32:07 2008'
>>> p = ...
'\w{3}\s\w{3}\s\d{2}\s\d{2}:\d{2}:\d{2}\s\d{4}'
>>> re.match(p,now) != None
True
```

pattern	strings matched
<code>\w</code>	any alphanumeric character, same as <code>[A-Za-z]</code>
<code>\d</code>	any decimal digit, same as <code>[0-9]</code>
<code>\s</code>	any whitespace character
<code>re{n}</code>	n occurrences of <code>re</code>

Regular Expressions

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

pattern	strings matched
<code>literal</code>	strings starting with <code>literal</code>
<code>re1 re2</code>	strings starting with <code>re1</code> or <code>re2</code>

```
>>> from time import ctime
>>> now = ctime()
>>> now
'Fri Apr 25 07:32:07 2008'
>>> p = ...
'\w{3}\s\w{3}\s\d{2}\s\d{2}:\d{2}:\d{2}\s\d{4}'
>>> re.match(p,now) != None
True
```

pattern	strings matched
<code>\w</code>	any alphanumeric character, same as <code>[A-Za-z]</code>
<code>\d</code>	any decimal digit, same as <code>[0-9]</code>
<code>\s</code>	any whitespace character
<code>re{n}</code>	<code>n</code> occurrences of <code>re</code>

Matching 0 or 1 Occurrences

allow Ms ., Mr ., Mrs ., with or without the . (dot)

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the re module
matching strings with match()

Regular Expressions

common regular expression symbols
groups of regular expressions

```
>>> title = 'Mr?s?\.'? '
```

? matches 0 or 1 occurrences

. matches any character

\. matches the dot .

```
>>> re.match(title, 'Ms ') != None
True
```

```
>>> re.match(title, 'Ms. ') != None
True
```

```
>>> re.match(title, 'Miss ') != None
False
```

```
>>> re.match(title, 'Mr') != None
False
```

```
>>> re.match(title, 'Mr ') != None
True
```

```
>>> re.match(title, 'M ') != None
True
```

Matching 0 or 1 Occurrences

allow Ms ., Mr ., Mrs ., with or without the . (dot)

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the re module
matching strings with match()

Regular Expressions

common regular expression symbols
groups of regular expressions

```
>>> title = 'Mr?s?\.? '
```

? matches 0 or 1 occurrences

. matches any character

\. matches the dot .

```
>>> re.match(title, 'Ms ') != None
```

```
True
```

```
>>> re.match(title, 'Ms. ') != None
```

```
True
```

```
>>> re.match(title, 'Miss ') != None
```

```
False
```

```
>>> re.match(title, 'Mr') != None
```

```
False
```

```
>>> re.match(title, 'Mr ') != None
```

```
True
```

```
>>> re.match(title, 'M ') != None
```

```
True
```

Matching 0 or 1 Occurrences

allow Ms ., Mr ., Mrs ., with or without the . (dot)

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the re module
matching strings with match()

Regular Expressions

common regular expression symbols
groups of regular expressions

```
>>> title = 'Mr?s?\.? '
```

? matches 0 or 1 occurrences

. matches any character

\. matches the dot .

```
>>> re.match(title, 'Ms ') != None
```

```
True
```

```
>>> re.match(title, 'Ms. ') != None
```

```
True
```

```
>>> re.match(title, 'Miss ') != None
```

```
False
```

```
>>> re.match(title, 'Mr') != None
```

```
False
```

```
>>> re.match(title, 'Mr ') != None
```

```
True
```

```
>>> re.match(title, 'M ') != None
```

```
True
```

Matching 0 or 1 Occurrences

allow Ms ., Mr ., Mrs ., with or without the . (dot)

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the re module
matching strings with match()

Regular Expressions

common regular expression symbols
groups of regular expressions

```
>>> title = 'Mr?s?\.? '
```

? matches 0 or 1 occurrences

. matches any character

\. matches the dot .

```
>>> re.match(title, 'Ms ') != None
```

```
True
```

```
>>> re.match(title, 'Ms. ') != None
```

```
True
```

```
>>> re.match(title, 'Miss ') != None
```

```
False
```

```
>>> re.match(title, 'Mr') != None
```

```
False
```

```
>>> re.match(title, 'Mr ') != None
```

```
True
```

```
>>> re.match(title, 'M ') != None
```

```
True
```

Matching 0 or 1 Occurrences

allow Ms ., Mr ., Mrs ., with or without the . (dot)

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the re module
matching strings with match()

Regular Expressions

common regular expression symbols
groups of regular expressions

```
>>> title = 'Mr?s?\.? '
```

? matches 0 or 1 occurrences

. matches any character

\. matches the dot .

```
>>> re.match(title, 'Ms ') != None
True
```

```
>>> re.match(title, 'Ms. ') != None
True
```

```
>>> re.match(title, 'Miss ') != None
False
```

```
>>> re.match(title, 'Mr') != None
False
```

```
>>> re.match(title, 'Mr ') != None
True
```

```
>>> re.match(title, 'M ') != None
True
```

Matching 0 or 1 Occurrences

allow Ms ., Mr ., Mrs ., with or without the . (dot)

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the re module
matching strings with match()

Regular Expressions

common regular expression symbols
groups of regular expressions

```
>>> title = 'Mr?s?\.? '
```

? matches 0 or 1 occurrences

. matches any character

\. matches the dot .

```
>>> re.match(title, 'Ms ') != None
```

```
True
```

```
>>> re.match(title, 'Ms. ') != None
```

```
True
```

```
>>> re.match(title, 'Miss ') != None
```

```
False
```

```
>>> re.match(title, 'Mr') != None
```

```
False
```

```
>>> re.match(title, 'Mr ') != None
```

```
True
```

```
>>> re.match(title, 'M ') != None
```

```
True
```

Character Classes

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the re module
matching strings with match()

Regular Expressions

common regular expression symbols
groups of regular expressions

Match with specific characters.

A name has to start with upper case:

```
>>> name = '[A-Z][a-z]*'
>>> G = 'Guido van Rossum'
>>> re.match(name,G)
>>> _.group()
'Guido'

>>> g = 'guido'
>>> re.match(name,g) == None
True
```

Character Classes

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the re module
matching strings with match()

Regular Expressions

common regular expression symbols
groups of regular expressions

Match with specific characters.

A name has to start with upper case:

```
>>> name = '[A-Z][a-z]*'
>>> G = 'Guido van Rossum'
>>> re.match(name,G)
>>> _.group()
'Guido'

>>> g = 'guido'
>>> re.match(name,g) == None
True
```

Character Classes

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the re module
matching strings with match()

Regular Expressions

common regular expression symbols
groups of regular expressions

Match with specific characters.

A name has to start with upper case:

```
>>> name = '[A-Z][a-z]*'
>>> G = 'Guido van Rossum'
>>> re.match(name,G)
>>> _.group()
'Guido'

>>> g = 'guido'
>>> re.match(name,g) == None
True
```

Counting Words Pattern Matching

Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

1 Dictionaries

free books as .txt files
dictionaries in Python
sorting dictionary items

2 Pattern Matching

using the `re` module
matching strings with `match()`

3 Regular Expressions

common regular expression symbols
groups of regular expressions

the `groups()` method

Dictionaries

free books as .txt files

dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols

groups of regular expressions

Groups of regular expressions are designated with parenthesis, between `(` and `)`.

Syntax:

```
< pattern > = ( < group1 > ) ( < group2 > )  
m = re.match( < pattern > , < string > )  
if m != None: m.groups()
```

After a successful match, `groups()` returns a tuple of those parts of the string that matched the pattern.

the `groups()` method

Dictionaries

free books as .txt files

dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols

groups of regular expressions

Groups of regular expressions are designated with parenthesis, between (and).

Syntax:

```
< pattern > = ( < group1 > ) ( < group2 > )  
m = re.match( < pattern > , < string > )  
if m != None: m.groups()
```

After a successful match, `groups()` returns a tuple of those parts of the string that matched the pattern.

the `groups()` method

Dictionaries

free books as .txt files

dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols

groups of regular expressions

Groups of regular expressions are designated with parenthesis, between `(` and `)`.

Syntax:

```
< pattern > = ( < group1 > ) ( < group2 > )  
m = re.match( < pattern > , < string > )  
if m != None: m.groups()
```

After a successful match, `groups()` returns a tuple of those parts of the string that matched the pattern.

Extracting hours, seconds, minutes

using the `groups()` method

Dictionaries

free books as .txt files

dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols

groups of regular expressions

```
>>> from time import ctime
>>> now = ctime()
>>> now
'Fri Apr 25 07:32:07 2008'
>>> t = now.split(' ')[3]
>>> t
'07:32:07'
>>> format = '(\d\d):(\d\d):(\d\d)'
>>> m = re.match(format,t)
>>> m.groups()
('07', '32', '07')
>>> (hours, minutes, seconds) = _
>>> minutes
'32'
```

Extracting hours, seconds, minutes

using the `groups()` method

Dictionaries

free books as .txt files

dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols

groups of regular expressions

```
>>> from time import ctime
>>> now = ctime()
>>> now
'Fri Apr 25 07:32:07 2008'
>>> t = now.split(' ')[3]
>>> t
'07:32:07'
>>> format = '(\d\d):(\d\d):(\d\d)'
>>> m = re.match(format,t)
>>> m.groups()
('07', '32', '07')
>>> (hours, minutes, seconds) = _
>>> minutes
'32'
```

Extracting hours, seconds, minutes

using the `groups()` method

Dictionaries

free books as .txt files

dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols

groups of regular expressions

```
>>> from time import ctime
>>> now = ctime()
>>> now
'Fri Apr 25 07:32:07 2008'
>>> t = now.split(' ')[3]
>>> t
'07:32:07'
>>> format = '(\d\d):(\d\d):(\d\d)'
>>> m = re.match(format,t)
>>> m.groups()
('07', '32', '07')
>>> (hours, minutes, seconds) = _
>>> minutes
'32'
```

Extracting hours, seconds, minutes

using the `groups()` method

Dictionaries

free books as .txt files

dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols

groups of regular expressions

```
>>> from time import ctime
>>> now = ctime()
>>> now
'Fri Apr 25 07:32:07 2008'
>>> t = now.split(' ')[3]
>>> t
'07:32:07'
>>> format = '(\d\d):(\d\d):(\d\d)'
>>> m = re.match(format,t)
>>> m.groups()
('07', '32', '07')
>>> (hours, minutes, seconds) = _
>>> minutes
'32'
```

Extracting hours, seconds, minutes

using the `groups()` method

Dictionaries

free books as .txt files

dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols

groups of regular expressions

```
>>> from time import ctime
>>> now = ctime()
>>> now
'Fri Apr 25 07:32:07 2008'
>>> t = now.split(' ')[3]
>>> t
'07:32:07'
>>> format = '(\d\d):(\d\d):(\d\d)'
>>> m = re.match(format,t)
>>> m.groups()
('07', '32', '07')
>>> (hours, minutes, seconds) = _
>>> minutes
'32'
```

Summary and Exercises

Dictionaries

free books as .txt files

dictionaries in Python
sorting dictionary items

Pattern Matching

using the `re` module
matching strings with `match()`

Regular Expressions

common regular expression symbols
groups of regular expressions

Read Chapter 8 of *Python Programming in Context*.

- 1 Words may contain `\n` or other special symbols. Modify the code to first strip the word of special symbols and to convert to lower case before updating the dictionary.
- 2 Modify the script `wordswardict.py` to count letters instead of words.
- 3 Download of 2 different authors 2 different texts from `www.gutenberg.org`. Do the word frequencies tell which texts are written by the same author?
- 4 Write a regular expression to match all words that start with `a` and end with `t`.
- 5 Modify `wordswardict.py` so that it prompts the user for a regular expression and then builds a frequency table for those words that match the regular expression.