More on Python

Mirco Kocher
mirco.kocher@unine.ch

September 28th, 2016

Outline

- References
- File IO
- Regular expressions
- Problem Set 02

References

```
>>> a = [1,2,3]
>>> b = a
>>> a.append(4)
>>> a, b
```

References

```
>>> a = [1,2,3]
>>> b = a          # new reference to same list object
>>> a.append(4)
>>> a, b
([1, 2, 3, 4], [1, 2, 3, 4])
>>> c = a[:2]
>>> a, b, c
```

References

```
>>> a = [1,2,3]
>>> b = a          # new reference to same list object
>>> a.append(4)
>>> a, b
([1, 2, 3, 4], [1, 2, 3, 4])
>>> c = a[:2]     # new list object and copy first two elements
>>> a, b, c
([1, 2, 3, 4], [1, 2, 3, 4], [1, 2])
>>> d = a[:]
>>> a.append(5)
>>> a, b, c, d
```

References

```
>>> a = [1,2,3]
>>> b = a          # new reference to same list object
>>> a.append(4)
>>> a, b
([1, 2, 3, 4], [1, 2, 3, 4])
>>> c = a[:2]     # new list object and copy first two elements
>>> a, b, c
([1, 2, 3, 4], [1, 2, 3, 4], [1, 2])
>>> d = a[:]      # new list object and copy all elements
>>> a.append(5)
>>> a, b, c, d
([1, 2, 3, 4, 5], [1, 2, 3, 4, 5], [1, 2], [1, 2, 3, 4])
```

Outline

- References
- File IO
- Regular expressions
- Problem Set 02

IO

```
>>> inputFile = open('D:\\names.txt', 'r')
>>> outputFile = open('D:\\shortNames.txt', 'w')
>>> for line in inputFile:
        if len(line) < 5:
            bytesWritten = outputFile.write(line)
>>> inputFile.close()
>>> outputFile.close()
```

IO

```
>>> inputFile = open('D:\\names.txt', 'r')
>>> outputFile = open('D:\\shortNames.txt', 'w')
>>> for line in inputFile:
        if len(line) < 5:
            bytesWritten = outputFile.write(line)
>>> inputFile.close()
>>> outputFile.close()
```

Python 2: write() returns no value
bytesWritten would be None
outputFile.write(line)

IO

- with-block ensures that a resource is cleaned up

```
>>> with open('D:\\names.txt', 'r') as inputFile:
        with open('D:\\names2.txt', 'w') as outputFile:
            for line in inputFile:
                bytesWritten = outputFile.write(line)
```

IO

- with-block ensures that a resource is cleaned up

```
>>> with open('D:\\names.txt', 'r') as inputFile:
        with open('D:\\names2.txt', 'w') as outputFile:
            for line in inputFile:
                bytesWritten = outputFile.write(line)
```

- You *should* close files using close() or with-block
  - Shows that you care, good practice
  - Removed from memory, all written to file

IO

- with-block ensures that a resource is cleaned up

```
>>> with open('D:\\names.txt', 'r') as inputFile:
        with open('D:\\names2.txt', 'w') as outputFile:
            for line in inputFile:
                bytesWritten = outputFile.write(line)
```

- You *should* close files using close() or with-block
  - Shows that you care, good practice
  - Removed from memory, all written to file
- You don't *have* to close files
  - Python will do it at program exit
  - Python will eventually do it when it is garbage collected

IO

```
>>> with open('D:\\names.txt', 'r') as inputFile:
        text = inputFile.read()
>>> with open('D:\\names2.txt', 'w') as outputFile:
        bytesWritten = outputFile.write(text)
```

Outline

- References
- File IO
- Regular expressions
- Problem Set 02

Regular expressions

- Pattern to search in text similar to wildcards
- Wildcard for all text files:

  *.txt
- Regex for all text files:

  .*\.txt$
- re module in Python

```
>>> import re
>>> re.search(pattern, string)          # returns object or None
>>> re.findall(pattern, string)         # returns list
>>> re.sub(pattern, replace, string)    # returns new string
```

Regular expressions

- Match a string

```
>>> t = 'Lorem ipsum dolor sit amet, consectetur adipiscing elit.'
>>> re.search(r'or', t)
<_sre.SRE_Match object; span=(1, 3), match='or'>
>>> re.findall(r'or', t)
['or', 'or']
>>> re.sub(r'or', 'OR', t)
'LORem ipsum dolOR sit amet, consectetur adipiscing elit.'
```

Regular expressions

- Match strings

```
>>> t = 'Lorem ipsum dolor sit amet, consectetur adipiscing elit.'
>>> re.search(r'[sl]it', t)
<_sre.SRE_Match object; span=(18, 21), match='sit'>
>>> re.findall(r'[sl]it', t)
['sit', 'lit']
>>> re.sub(r'[sl]it', '', t)
'Lorem ipsum dolor  amet, consectetur adipiscing e.'
```

Regular expressions

- Match string groups

```
>>> t = 'Lorem ipsum dolor sit amet, consectetur adipiscing elit.'
>>> re.search(r' (s|el)it', t)
<_sre.SRE_Match object; span=(17, 21), match=' sit'>
>>> re.findall(r' (s|el)it', t)
['s', 'el']
>>> re.findall(r'( (s|el)it)', t)
[(' sit', 's'), (' elit', 'el')]
>>> re.sub(r' (s|el)it', '', t)
'Lorem ipsum dolor amet, consectetur adipiscing.'
```

Regular expressions

- Match string groups

  '(?:)' Don’t capture this group

```
>>> t = 'Lorem ipsum dolor sit amet, consectetur adipiscing elit.'
>>> re.findall(r' (?:s|el)it', t)
[' sit', ' elit']
```

Regular expressions

- Match a string with variable length

  'o{1}' matches exactly one o (same as 'o')

  'o{1,}' matches one o or more (same as 'o+' or 'oo*')

  'o{,1}' matches zero or one o (same as 'o?')

```
>>> t = 'Hellooo over there'
>>> re.search(r'Hello{2,5}', t)
<_sre.SRE_Match object; span=(0, 7), match='Hellooo'>
>>> re.findall(r'Hello{2,5}', t)
['Hellooo']
>>> re.sub(r'Hello{2,5}', 'Hello', t)
'Hello over there'
```

Regular expressions

- Match a string with variable length

  'o{,1}' matches **zero** or one o (same as 'o?')

```
>>> re.findall(r'o?', 'Hello')
```

Regular expressions

- Match a string with variable length

  'o{,1}' matches **zero** or one o (same as 'o?')

```
>>> re.findall(r'o?', 'Hello')
['', '', '', '', 'o', '']
>>> re.sub(r'o?', 'X', 'Hello')
```

Regular expressions

- Match a string with variable length

  'o{,1}' matches **zero** or one o (same as 'o?')

```
>>> re.findall(r'o?', 'Hello')
['', '', '', '', 'o', '']
>>> re.sub(r'o?', 'X', 'Hello')
'XHXeXlXlX'
```

Regular expressions

- Match a special string

```
>>> t = 'Lorem ipsum dolor sit amet, consectetur adipiscing elit.'
>>> re.search(r'\.', t)
<_sre.SRE_Match object; span=(55, 56), match='.'>
>>> re.findall(r'\.', t)
['.']
>>> re.sub(r'\.', '!', t)
'Lorem ipsum dolor sit amet, consectetur adipiscing elit!'
```

- Characters to escape with backslash:

  \ ^ $. | ? * + ( ) [ ] {

Regular expressions

- Match one character of a group

  '.' for any character (except the newline)

  '[a-z]' for any lowercase letter from a to z

  '[M-Q]' for any uppercase letter from M to Q

  '\d' for any digit (same as '[0-9]')

  '\s' for any whitespace (like space, tab, newline)

  '\w' for any word character (letters, digits, or underscore)

Regular expressions

- Don't match one character of a group

  '[^A-Ef-k5-9]' for any non-uppercase letter from A to E and non-lowercase letter from f to k and no digit from 5 to 9

  '\D' for any non-digit (same as '[^0-9]')

  '\S' for any non-whitespace

  '\W' for and non-word character

Regular expressions

- Match one character with an anchor

  '^' for the start of the string or line

  '$' for the end of the string or line

  '\b' for the word boundary (whitespace or punctuation)

```
>>> re.findall(r'^b', 'bob')
['b']                                   # only the first b
>>> re.findall(r'b$', 'bob')
['b']                                   # only the last b
>>> re.findall(r'\bOK\b', 'OK, OK OKOK OK1 (OK) OK.')
['OK', 'OK', 'OK', 'OK']                # not the OK in OKOK or OK1
```

Regular expressions

- Match lazy instead of greedy

```
>>> re.findall(r'<.+>', 'This is a <em>tag</em> test')
```

Regular expressions

- Match lazy instead of greedy

```
>>> re.findall(r'<.+>', 'This is a <em>tag</em> test')
['<em>tag</em>']
```

Regular expressions

- Match lazy instead of greedy

```
>>> re.findall(r'<.+>', 'This is a <em>tag</em> test')
['<em>tag</em>']
>>> re.findall(r'<.+?>', 'This is a <em>tag</em> test')
['<em>', '</em>']
>>> re.findall(r'\bthe.*\b', 'So this was there for the 24h')
```

Regular expressions

- Match lazy instead of greedy

```
>>> re.findall(r'<.+>', 'This is a <em>tag</em> test')
['<em>tag</em>']
>>> re.findall(r'<.+?>', 'This is a <em>tag</em> test')
['<em>', '</em>']
>>> re.findall(r'\bthe.*\b', 'So this was there for the 24h')
['there for the 24h']
>>> re.findall(r'\bthe.*?\b', 'So this was there for the 24h')
['there', 'the']
```

Regular expressions

Pattern	Text	Match?
r'^[a-z][a-z0-9_-]{3,8}$'	'a-us3r_' 'my-us3r_n4m3' '4-us3r'	

Regular expressions

Pattern	Text	Match?
r'^[a-z][a-z0-9_-]{3,8}$'	'a-us3r_' 'my-us3r_n4m3' '4-us3r'	Yes No No
r'^#?([a-f0-9]{6}|[a-f0-9]{3})$'	'#a3c113' '#4d82h4' 'abc'	

Regular expressions

Pattern	Text	Match?
r'^[a-z][a-z0-9_-]{3,8}$'	'a-us3r_' 'my-us3r_n4m3' '4-us3r'	Yes No No
r'^#?([a-f0-9]{6}|[a-f0-9]{3})$'	'#a3c113' '#4d82h4' 'abc'	Yes No Yes
r'^(?:(?:25[0-5]|2[0-4][0-9]| [0-1]?[0-9]?[0-9])\.?){4}$'	'255.256.1.1' '192.168.1.1' '19216811' '0.0..127'	

Regular expressions

Pattern	Text	Match?
r'^[a-z][a-z0-9_-]{3,8}$'	'a-us3r_' 'my-us3r_n4m3' '4-us3r'	Yes No No
r'^#?([a-f0-9]{6}|[a-f0-9]{3})$'	'#a3c113' '#4d82h4' 'abc'	Yes No Yes
r'^(?:(?:25[0-5]|2[0-4][0-9]| [0-1]?[0-9]?[0-9])\.?){4}$'	'255.256.1.1' '192.168.1.1' '19216811' '0.0..127'	No Yes Yes No
r'^(?:XC|XL|L?X{0,3})(?:IX|IV|V?I{0,3})$'	'XCIX' 'III' '' 'IC' 'XII' 'XIIV'	

Regular expressions

Pattern	Text	Match?
r'^[a-z][a-z0-9_-]{3,8}$'	'a-us3r_' 'my-us3r_n4m3' '4-us3r'	Yes No No
r'^#?([a-f0-9]{6}|[a-f0-9]{3})$'	'#a3c113' '#4d82h4' 'abc'	Yes No Yes
r'^(?:(?:25[0-5]|2[0-4][0-9]| [0-1]?[0-9]?[0-9])\.?){4}$'	'255.256.1.1' '192.168.1.1' '19216811' '0.0..127'	No Yes Yes No
r'^(?:XC|XL|L?X{0,3})(?:IX|IV|V?I{0,3})$'	'XCIX' 'III' '' 'IC' 'XII' 'XIIV'	Yes Yes Yes No Yes No

Regular expressions

- http://www.regexpal.com/

Questions?

Outline

- References
- File IO
- Regular expressions
- Problem Set 02

Problem Set 02

- Handle exceptions
- Read & write files
- User regular expressions

- Deadline: October 5th, 2016 at 9:00 AM

Questions?