Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Introduction to Programming with Python

II.6. Advanced topics

Oscar Delgado
oscar.delgado@uam.es

Pattern Matching: regular expressions

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Regular expressions

A **regular expression** is special sequence of characters for matching or finding other strings

Pattern	Target	Match position	Search position
GAATTC	GAATTC	0	0
	TTGAATTC	None	2
	AATGTGAATTCT	None	5

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Regular expressions

Now imagine that '?' means a match with any character

Pattern	Target	Match position	Search position
G??TTC	GAATTC	0	0
	GATTTC	0	0
	GACTTC	0	0
	...	0	0
	GTTTTC	0	0

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Character sets

A **character set** consists of one or more characters enclosed in square brackets

There is a match is string contains ONLY one character from the character set

'CC[GA][TC]GG'

CC**GT**GG
CC**GC**GG
CC**AT**GG
CC**AC**GG

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Character sets

We can give range of values – for example, [a-j] for the letters ‘a’ through ‘j’

Range	Description
[A-Z]	any capital letter
[0-9]	any digit
[A-Za-z0-9]	any letter or digit

Character classes

Character classes are simply convenient abbreviations for commonly used character sets

Character	Matches
\d	Any digit
\D	Any nondigit
\s	Any whitespace character
\S	Any nonwhitespace character
\w	Any character considered part of a word
\W	Any character not considered part of a word

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Repetition

Examples

Character	Matches
?	Zero or one repetitions of the preceding regular expression
*	Zero or more repetitions of the preceding regular expression
+	One or more repetitions of the preceding regular expression
{*n*}	Exactly *n* repetitions of the preceding regular expression
{*m*,*n*}	Between *m* and *n* (inclusive) repetitions of the preceding regular expression

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Boundaries

Notation to indicate match only when the pattern is at beginning of end of the string

Character	Matches
^	The start of a line or the beginning of the pattern
$	The end of a line or the end of the pattern

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Boundaries

Example

Pattern	Matches
^CG	A target line or string starting with CG
TATATA$	A target line or string ending with TATATA

Summary

Expression	Description
`[]`	any of the characters inside the brackets
`.`	any character except the newline
`+`	1 or more of the preceding
`*`	0 or more of the preceding
`?`	0 or 1 of the preceding
`{m}`	exactly `m` of the preceding
`{m,n}`	between `m` and `n` (inclusive) of the preceding
`?`	following `+`, `*`, `?`, `{m}`, and `{m,n}` — take as few as possible
`\`	escape special characters
`|`	"or"
`^`	(at start of pattern) match just the first occurrence
`$`	(at end of pattern) match just the last occurrence
`\d \D`	any digit (non-digit)
`\w \W`	any letter or number (non-letter or -number)
`\s \S`	any whitespace (non-whitespace)

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Python pattern matching

Basic methods

- **replace()** method of strings is used to replace all occurrences of one string with another

```
'abcdef abcxyz'.replace('abc', '*')
```

```
*def *xyz
```

- **index()** method is used to find the first occurrence of a substring in a string

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Python pattern matching

Regular expressions

- **sub()** functions works as follows:

```
import re

re.sub(pattern, replacement, string)
```

This searches through *string* for *pattern* and replaces anything matching that pattern with the string *replacement*.

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Python regular expressions

Example

- Replace all ocurrences of ‘A’ and ‘G’ with ‘*’ in the following string:

ATGCCGTAACTGCA

```
import re

re.sub(‘[AG]’,’*’,’ ATGCCGTAACTGCA’)
```

Python regular expressions

findall() The findall() function returns a list of all the matches found

```
re.findall(r'[AB]\d', 'A3 + B2 + A9')
```

```
['A3', 'B2', 'A9']
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Python regular expressions

match(), search() These are useful if you just want to know if a match occurs.

The difference between these two functions is **match()** only checks to see if the beginning of the string matches the pattern, while **search()** searches through the string until it finds a match

Python regular expressions

Example

```
if (re.match(r'ZZZ', 'abc ZZZ xyz')):
    print('Match found at beginning.')
else:
    print('No match at beginning')

if (re.search(r'ZZZ', 'abc ZZZ xyz')):
    print('Match found in string.')
else:
    print('No match found.')
```

```
No match at beginning.
Match found in string.
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Lambda functions

Functional programming

Lambda functions Small anonymous functions, without a name. These functions are throw-away functions: they are just used where they have been created.

General expresion

```
lambda argument_list: expresion
```

```
sum = lambda x,y: x + y
print(sum(2,3))
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Lambda functions

```
sum = lambda x,y: x + y
print(sum(2,3))
```

```
def sum(x,y):
    return x+y

print(sum(2,3))
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Lambda functions

Lambda functions are mainly used in combination with the functions **filter()**, **map()** and **reduce()**.

The lambda feature was added to Python due to the demand from Lisp programmers (functional programming).

filter() function

The function **filter(f, list)** offers an elegant way to filter out all the elements of a list, for which the function *f* returns True.

```
fib = [0,1,1,2,3,5,8,13,21,34,55]
result = filter(lambda x: x % 2 == 0, fib)
print result

[0, 2, 8, 34]
```

Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

map() function

map(func, list) applies the function *func* to all the elements of the list *list*. It returns a new list with the elements changed by *func*

```
lst = [0,1,2,3,4,5,6]
result = map(lambda x: x+1, lst)
print result

[1,2,3,4,5,6,7]
```

Escuela Politécnica Superior

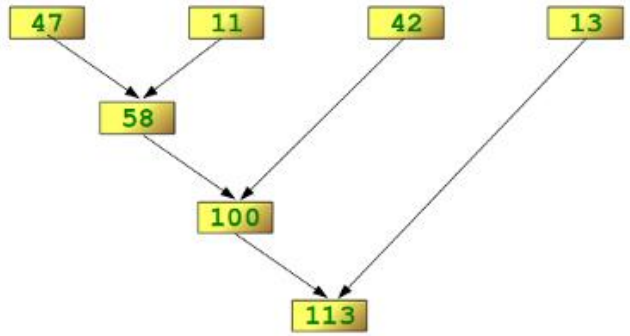
UAM UNIVERSIDAD AUTÓNOMA DE MADRID

reduce() function

The function **reduce**(func, lst) continually applies the function *func()* to the sequence *lst* by pairs, till only one element is left.

```
print(reduce(lambda x,y: x+y, [47,11,42,13]))
```

```
113
```



Escuela Politécnica Superior

UAM UNIVERSIDAD AUTÓNOMA DE MADRID

Summary

Function	Description
filter(f,l)	Apply f to each element i of l and returns i only if f(i) = True
map(f,l)	Apply f to each element i of l and returns a new list with f(i)
reduce(f,l)	Apply f to elements of l by pairs, till only one element is left