Python Tutorial

Lambda

20180809 이승혁

Index

I. Lambda

II. map

III. reduce

IV. filter

Lambda

```
lambda 인자 : 표현식
```

```
def sum(a, b):
    return int(a) + int(b)

print "Sum :", sum(a, b)
print "Lambda :", (lambda x, y : x + y)(a, b)
```

```
Input two numbers to add.
-> 1 2

Sum : 3
Lambda : 3
```

Lambda

- Advantages

I. Simple code

II. Efficient use

- Disadvantages

I. Slow

II. Hard to understand

Map

```
map(함수, 리스트)

inp = raw_input().split()

inp = map(int, inp)
```

```
1 2 3 4 5
49 <type 'str'>
50 <type 'str'>
51 <type 'str'>
52 <type 'str'>
53 <type 'str'>
1 <type 'int'>
2 <type 'int'>
3 <type 'int'>
4 <type 'int'>
5 <type 'int'>
```

Map

```
inp = map(lambda x : math.pow(float(x), float(x)),
raw_input().split())
```

```
1 2 3
[1.0, 4.0, 27.0]
```

Reduce

reduce(함수, 리스트)

```
def mulsum(a):
        sum = 0
        for i in range(len(inp)):
                sum += inp[i]

        retrun sum

print "Sum :", mulsum(inp)
print "Reduce :", (reduce(lambda x, y : x + y, inp))
```

Reduce

```
Input any numbers to add.
-> 1 2 3 4 5 6 7  8 9 10

List : [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Sum : 55
Reduce : 55
```

Filter

```
filter(함수, 리스트)
```

```
def evenfinder(inp):
        res = []
        for i in range(len(inp)):
                if inp[i] % 2 == 0:
                        res.append(inp[i])
        return res

print "Function :", evenfinder(inp)
print "Filter :", filter(lambda x : x % 2 == 0, inp)
```

Filter

```
Input any number to check even.
->  1 2 3 4 5 6 7 8 9 10

Function : [2, 4, 6, 8, 10]
Filter : [2, 4, 6, 8, 10]
List Expression : [2, 4, 6, 8, 10]
```

Filter?

```
Filter : [2, 4, 6, 8, 10]
List Expression : [2, 4, 6, 8, 10]
```

```
filter(lambda x : x % 2 == 0, inp)

[i for i in inp if i % 2 == 0]
```

Q?

INDEX

I. Regular Expressions
II. Meta Characters
III. Functions
IV. Match Object

I. Regular Expression

REGULAR  EXPRESSIONS

python

패턴	설명	예제
^	이 패턴으로 시작해야 함.	^abc : abc로 시작해야 함. abcd, abc12 등
$	이 패턴으로 종료되어야 함.	xyz$: xyz로 종료되어야 함. 12xyz, strxyz 등
[문자들]	문자 클래스로 대괄호 안의 문자들중에 하나여야 함. 가능한 문자들의 집합을 정의함.	[Pp]ython : "Python" 혹은 "python"
[^문자들]	[문자들]의 반대로 피해야 할 문자들의 집합을 정의함.	[^aeiou] : 소문자 모음이 아닌 문자들
|	두 패턴 중 하나여야 함	a | b : a 또는 b여야 함
?	앞 패턴이 없거나 하나여야 함. (optional 패턴을 정의할 때 사용)	\d? : 숫자가 하나 있거나 없어야 함
+	앞 패턴이 하나 이상이어야 함.	\d+ : 숫자가 하나 이상이어야 함
*	앞 패턴이 0개 이상이어야 함.	\d* : 숫자가 없거나 하나 이상이어야 함
패턴{n}	앞 패턴이 n번 반복되는 경우.	\d{3} : 숫자가 세번 반복되어야 함
패턴{n, m}	앞 패턴이 최소 n번에서 최대 m번 반복되는 경우.	\d{3, 5} : 숫자가 세번, 네번 혹은 다섯번 반복되어야 함.
\d	숫자 0~9. [0-9]와 동일.	\d\d\d : 0~9 범위의 숫자 세 개를 의미. 123, 000 등
\w	문자와 숫자를 의미(Alphanumeric). [a-zA-Z0-9]와 동일.	\w\w\w : 문자 세 개를 의미. abc, ABC 등
\s	공백문자를 의미하며 [\t\n\r\f]와 동일. 앞의 빈 칸 하나는 스페이스를 의미함.	\s\s : 공백문자 두개를 의미. (' ', '\t', '\n', '\r', '\f')
\D, \W, \S	대문자의 경우 소문자 패턴의 반대를 뜻함.	\D : [^0-9]를 의미.
.	\n을 제외한 모든 문자를 의미.	.{3} : 문자 세개를 의미. abc, 123 등

III. Functions

Method	목적
match()	문자열의 처음부터 정규식과 매칭되는지 조사한다.
search()	문자열 전체를 검색하여 정규식과 매칭되는지 조사한다.
findall()	정규식과 매치되는 모든 문자열(substring)을 리스트로 리턴한다.
finditer()	정규식과 매치되는 모든 문자열(substring)을 iterator 객체로 리턴한다.

```
if f1:
    if i.split("-")[2] != date:
        list1.append("\nDay : %s" % i.split("-")[2])
    date = i.split("-")[2]
```

IV. Match Object

Method	목적
group()	매치된 문자열을 리턴한다.
start()	매치된 문자열의 시작 위치를 리턴한다.
end()	매치된 문자열의 끝 위치를 리턴한다.
span()	매치된 문자열의 (시작, 끝) 에 해당되는 튜플을 리턴한다.

```
m = p.search("3 python")

print m.group()
print m.start()
print m.end()
print m.span()
```

```
2018-02-01 16:55:30,750: Key.up
2018-02-01 16:55:31,279: Key.down
2018-02-01 16:55:42,262: Key.cmd
2018-02-01 16:55:42,262: u'r'
2018-02-01 16:55:42,901: Key.enter
2018-02-01 16:55:44,150: u'd'
2018-02-01 16:55:44,275: u'i'
2018-02-01 16:55:44,398: u'r'
2018-02-01 16:55:44,555: Key.enter
2018-02-01 16:55:49,032: u'c'
2018-02-01 16:55:49,173: u'd'
2018-02-01 16:55:49,266: Key.space
2018-02-01 16:55:49,546: u'd'
2018-02-01 16:55:49,750: u'e'
2018-02-01 16:55:49,983: u's'
2018-02-01 16:55:50,686: Key.tab
2018-02-01 16:55:50,997: Key.enter
2018-02-01 16:55:52,385: u'd'
2018-02-01 16:55:52,510: u'i'
2018-02-01 16:55:52,635: u'r'
2018-02-01 16:55:52,792: Key.enter
2018-02-01 16:55:53,884: u'c'
2018-02-01 16:55:54,009: u'd'
2018-02-01 16:55:54,148: Key.space
2018-02-01 16:55:56,832: Key.shift
2018-02-01 16:55:57,035: u'T'
2018-02-01 16:55:58,142: u'r'
2018-02-01 16:55:58,252: u'o'
2018-02-01 16:55:58,657: u'f'
2018-02-01 16:55:58,782: u'a'
2018-02-01 16:55:58,907: u'n'
2018-02-01 16:55:59,109: Key.enter
2018-02-01 16:56:00,124: Key.up
2018-02-01 16:56:00,372: Key.left
2018-02-01 16:56:00,513: Key.left
2018-02-01 16:56:00,684: Key.left
2018-02-01 16:56:00,904: Key.right
2018-02-01 16:56:01,371: Key.backspace
2018-02-01 16:56:01,605: u'j'
```

```
2018-02-08 04:19:09,151: u'r'
2018-02-08 04:19:09,246: u'e'
2018-02-08 04:19:09,704: u'v'
2018-02-08 04:19:09,950: u'e'
2018-02-08 04:19:10,082: u'r'
2018-02-08 04:19:10,624: u's'
2018-02-08 04:19:10,812: u'i'
2018-02-08 04:19:11,020: u'n'
2018-02-08 04:19:11,177: u'g'
2018-02-08 04:19:11,476: u'.'
2018-02-08 04:19:11,740: u'k'
2018-02-08 04:19:11,898: u'r'
2018-02-08 04:19:12,872: Key.enter
2018-02-08 04:19:21,052: u'l'
2018-02-08 04:19:21,101: u's'
2018-02-08 04:19:21,292: u'h'
2018-02-08 04:19:21,585: u'7'
2018-02-08 04:19:21,900: u'4'
2018-02-08 04:19:22,142: u'5'
2018-02-08 04:19:22,279: Key.tab
```

```
---------------Day : 01
---------------Time : 55

up(1)
r
cmd(1)
dir
cd
enter(2)
des
space(1)
tab(1)
dir
cd
enter(2)
space(1)
Trofan
shift(1)
```

```
---------------Day : 08
---------------Time : 19

reversing.kr
delete(1)
lsh745
enter(1)
tab(1)
fposition
enter(1)
shift_r(1)
```

고양 원더스 시절

이름	정규식(ku-sik-jung)
생년월일	1990년 6월 17일
국적	대한민국
신체	178cm, 80kg
출신지	경기도 성남시
출신학교	희망대초-성일중-교토국제고 - 오사카학원대
포지션	포수
투타	우투좌타
프로입단	2015년 신인 드래프트 2차 4라운드 37순위
소속팀	하쿠와 빅토리즈[1](2013) 고양 원더스(2013~2014) LG 트윈스(2015~2017)

LG 트윈스 등번호 62번				
유재호(2014)	→	**정규식(2015)**	→	현역

Q?