Python 프로그래밍
(version 2)

2016.7.26~27

윤형기 (hky@openwith.net)

진행 순서

단계	주요 내용
Python 기초	Python 개요와 설치
	변수, 문장, 조건문과 Loop, 함수
	Data Structures (List/Dictionary/Tuple/Set)
	문자열 기초 (String, 파일 읽기, 정규식 표현)
	DB, Module, Package, OOP, Exception, Web, …
한글 텍스트 처리 (Text Processing)	개요
	형태소 분석
	nltk
기타	API 활용
	numpy, pandas, matplotlib, iPython
	마무리 (Python과 MR 프로그래밍)

I. Python 기초

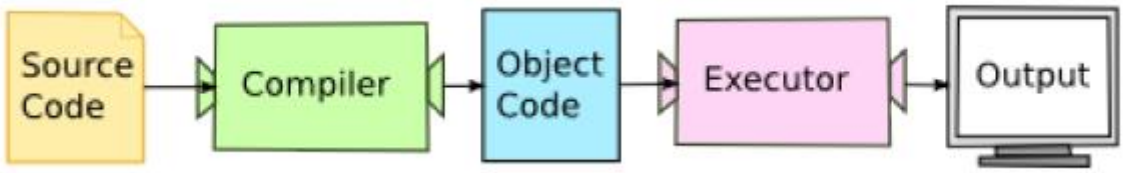
- 개요와 설치
- 데이터타입, 변수, 문장/ 조건문/ Loop
- 함수
- 데이터 구조
  - List
  - Dictionary
  - Tuple
  - Set
- String 처리
- 모듈

개요

- 일반론
  - 1990년대 후반 Guido van Rossum이 개발 (네덜란드)
  - 오픈소스 cross-platform
- Python의 특징 (1)
  - High-level 언어
  - Interpreted 언어
  - Interactive 언어
  - 객체지향 언어
  - Scripting 언어
- Python의 특징 (2)
  - 쉽다 (?)
    - Easy-to-read → Easy-to-maintain:
  - **수학적 개념, 함수형 언어, …**
  - 다른 언어와의 비교
    - C, C++, Java, R

- Python의 종류
  - "Python" or "CPython" written in C/C++
    - Version 2.7
    - Version 3.5
  - "Jython" ; Java for the JVM
  - "IronPython" ; in C# for the .Net environment
  - 분야별 Package 활용
- 개발환경
  - Python Interactive Shell – IDLE
  - 기타
    - vim, emacs
    - 오픈소스: Eclipse (PyDev), Komodo, ...
    - 상용: ...
  - iPython

- Python 이용방법
  - (1) 대화식 (Interactive Interpreter):
  - (2) 명령어 줄을 통한 Script 프로그램의 활용



  - 문서
    - http://www.python.org/
    - https://docs.python.org/

- 본 강의의 범위와 방식
  - 실습 코드와 데이터:
    - 별도 자료 참조

설치

- Python 기본설치
  - MS Windows 용 설치
  - Linux 설치
- 텍스트 분석 용 Python 패키지 설치 문제
  - 한글 형태소 분석 여부의 문제
  - 한글 사전/Thesaurus/Taxonomy 등의 활용 여부
  - NLTK
- 텍스트 분석 용 Framework의 검토
  - Solr, ELK, …

데이터 타입

- 기본 (built-in)
  - 숫자
    - Integers, Floats, Complex numbers
    - Booleans
  - String
    - Long string """ """, ''' '''
  - List
  - Tuple
  - Dictionary
  - Sets
  - File Objects
- User-defined
  - Object

변수

- 변수
  - …
- 변수명
  - 일반론
  - 보통 – PEP 규칙
    - Class는 대문자로 시작
    - 함수, Method 등은 camel type
    - Constant 는 모두 대문자
    - 기타변수는 소문자
  - Python Style Guide
    - http://legacy.python.org/dev/peps/pep-0008/

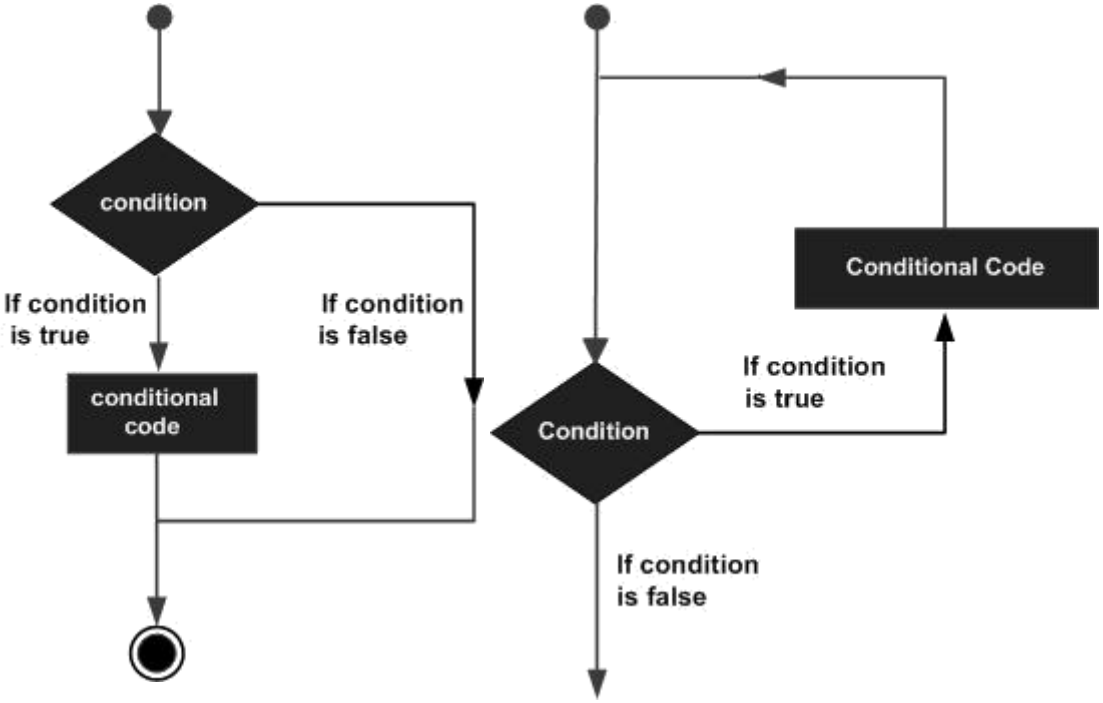
문장 (statements)

- 기본
  - Composition
  - 작업순서 (precedence)
    - (애매하면 괄호로 묶을 것)
- 사용자로부터의 입력

  >>> input()

조건문과 Loop문

- Block 지정
  - Indentation – Level 당 4 spaces
  - 조건지정: Boolean 값
- 조건문
  - if ~ elif ~ else
  - assertion
- Loop
  - while loop,
  - for loop
  - 다양한 iteration 방법 …
- List Comprehension
- 보충내용 (1)
  - pass, del, exec, del
- 보충내용 (2)
  - Sequence unpacking, …



조건지정: Boolean 값

- False
  - False, None, 0, "", {},(),[]
- True
  - True, False 아닌 기타의 모든 것

- 내부처리의 본질: True=1, False=0
  - **>>> True**
  - **True**
  - **>>> False**
  - **False**
  - **>>> True==1**
  - **True**
  - **>>> False==0**
  - **True**
  - **>>> True+False+50**
  - **51**

- True와 False는 bool type에 속함
  - **>>> type(True)**
  - **<class 'bool'>**
  - **>>> type(False)**
  - **<class 'bool'>**
  - **>>> bool('')**
  - **False**
  - **>>> bool(0)**
  - **False**

조건 실행

- if, elif, else
- Nested Blocks

```
var = 100
if var < 200:
   print("Expression value is less than
200")
   if var == 150:
      print("Which is 150")
   elif var == 100:
      print("Which is 100")
   elif var == 50:
      print("Which is 50")
elif var < 50:
   print("Expression value is less than 50")
else:
   print("Could not find true expression")

print("Good bye!")
```

- 비교연산자
  - ==
  - <, >
  - <=, >=
  - !=
  - is와 is not
  - in과 not in
- Equality 연산자
  - is
  - 양자를 구분할 것! – identity vs. equality

```
>>> x = y = [1,2,3]
>>> z = [1,2,3]
>>> x==y
True
>>> x==z
True
>>> x is y
True
>>> x is z
False
```

- Membership 연산자
  - in
- String 비교

- Boolean 연산자
  - Short-circuit Logic
  - …

Loop

- while loop

```
>>> x =1
>>> while x <=100:
        print(x)
        x+=1
```

- for loop
  - Code block 수행을 for each element of a set (or sequence, or other iterable object)
  - ☞ iterable object = iterate 가능한 객체

```
>>> words =
['this','is','a','wonder']
>>> for word in words:
        print(word)

this
is
a
wonder
```

- Range

```
>>> range(1,10,3)
range(1, 10, 3)
>>> print(range(1,10,3))
range(1, 10, 3)
>>> list(range(1,10,3))
[1, 4, 7]
```

  - # Python 2.x에서의 xrange는 3.x에서 range에 포함되었음

- Dictionary에 대한 iteration

- Iteration 관련 utilities
  - Parallel iteration

```
>>> # zip


# enumerate()로 index를 이용 가능.
>>> for i,v in enumerate(['tic','tac','toe']):
        print(i,v)


0 tic
1 tac
2 toe
```

- Reversed & Sorted iteration

- Loop 에서 나오는 방법
  - break
  - continue

  - while True/break idiom

  - while True는 무한반복이므로 문장내에 if /break 조건으로 처리

함수 (Functions)

- 개념
  - 함수?... 추상화의 한 단계
    - 함수, class, design patterns, ...
- 함수의 작성
  - def 문
- 함수에서의 Parameter
- Scoping의 문제
    - Local – 함수 내에서만 ...
    - Nonlocal –previously bound variable in the closest enclosing scope
    - Global – 함수 밖에 존재하고 이를 함수 밖에서 global 선언하여 access, 변경
- Recursion
- Lambda expression
  - In-line 정의하는 익명의 작은 함수 (단, return 문이 없음)
- Generator 함수
  - 자신이 원하는 iterator를 정의
  - yield 문 이용
- Decorator 함수
  - Function도 1st class로서 변수에 assign 되거나 parameter로서 전달될 수 있다.

- 함수의 작성
  - def – function을 정의

- Documenting

- Parameter
  - 용어
    - 정의 시: Formal parameter
    - 호출 시: Actual parameter = argument
  - Local 변수의 문제
    - local scope (자신이 지정된 function 또는 block)만을 이용하는 변수
  - 함수 내에서는 기본적으로 별도의 copy본을 만들어서 이용
    - 단, parameter object를 바꿀 경우 immutable에 유의
  - 종류
    - Positional parameter,
    - Keyword parameter,
    - default parameter

```
>>> def try_to_change(n):
        n="Me"

>>> name='You'
>>> try_to_change(name)
>>> name
'You'

>>> def change(n):
        n[0] = "Me"

>>> names = ["You","Her","Him"]
>>> change(names)
>>> names
['Me', 'Her', 'Him']
>>>
```

- Parameter의 개수변동

```
>>> def print_params(*params):
        print(params)


>>> print_params(1)
(1,)
>>> print_params(2,3,4)
(2, 3, 4)
>>> print_params('hk',
'john','obama')
('hk', 'john', 'obama')
>>>
```

- 여러 Keyword parameter 정보 수집
- 종합

- Scoping의 문제
  - Namespace란?
    - (변수 등의) 이름이 존재하는 곳
    - 일종의 invisible dictionary = scope
  - Nested scope
    - Function 안의 function – 예: using one function to create another.
- 재귀함수 (recursive functions)

```
# 예
# factorial의 정의: (a) 1의 factorial은 1
#   1 이상 n의 factorial은 n x factorial(n-1)
>>> def factorial(n):
        if n==1:
            return 1
        else:
            return n * factorial(n-1)
```

  - Local, nonlocal, global, 변수
    - Local – 함수 내에서만 …
    - Nonlocal – closest enclosing scope의 변수
    - Global –
  - Rebinding global variable
    - Making them refer to some new value
    - 변수를 함수 내에서 정의하는 순간 자동으로 local 변수가 됨.
      (다르게 지정할 수는 있음)

- Lambda 함수

```
>>> t2 = {'FtoK': lambda deg_f: 273 + (deg_f -32) * 5 / 9,
      'CtoK': lambda deg_c: 273 + deg_c}
>>> t2['FtoK'](32)
273.0
```

- Generator 함수

```
>>> def four():
        x = 0
        while x <4:
            print("in generator, x = ", x)
            yield x
            x+=1
```

```
>>> for i in four():
        print(i)

in generator, x =  0
0
in generator, x =  1
1
in generator, x =  2
2
in generator, x =  3
3
>>> 2 in four()
in generator, x =  0
in generator, x =  1
in generator, x =  2
True
```

- Decorators
  - Wrapping function (= decorator)
  - +
  - Wrapped function (작성
    - 후에 wrapping function의 argument로 이용)
    - @decorate
      함수정의()

파일 및 입출력

- 파일
  - 데이터를 일정한 방식으로 저장한 것.
- 파일 열기

  file object = open(file_name [, access_mode][, buffering])
  - File mode
  - Buffering
- 파일 관련 Methods
  - Read, Write
  - Pipe
  - Read, Write Lines
  - 파일 닫기
  - 주요 methods
- 파일 내용에 대한 Iteration (생략)
  - Byte 단위, Line 단위, 통째로 읽기
  - File Iterator

- 파일 열기
  - File mode
    - open()의 Mode argument

값	설명
'r'	읽기
'w'	쓰기
'a'	Append
'b'	Binary (다른 mode에 추가)
'+'	Read/write (다른 mode에 추가)

  - Binary 모드에서는 MS Windows에서의 ₩n₩r를 변환시키는 문제 등을 원천 해결

- Buffering
  - HDD 대신 메모리를 이용
    - 0 (False); unbuffered
    - 1 (True); buffered
    - 큰 숫자는 buffersize를 표시 (-1은 default buffer size를 의미)
- 표준 stream;
  - sys module
  - sys.stdin, sys.stdout, sys.stderr
- File-like
  - Streams
    - io module
    - (입출력의 3 가지 유형)
    - text I/O, binary I/O, raw I/O.
    - 각각에 대해 다양한 저장장치를 적용 - Concrete objects가 streams
  - urllib.urlopen
  - …

- 파일 관련 Methods
  - 파일 읽기 및 쓰기

- 코드 실습 (1)

```
>>> f = open('somefile.txt','w')
>>> f.write('hello, ')
7
>>> f.write('World')
5
>>> f.close()
>>>
>>> f=open('somefile.txt','r')
>>> f.read(4)
'hell'
>>> f.read()
'o, World'
```

- 코드 실습 (2)

모듈

- Module은 확장기능(을 담은 파일)
  - Attribute와 dot operator
  - namespace의 문제
  - Scope와 lookup 규칙
- 제공되는 모듈
  - time module
  - math module
- 사용자 작성 모듈

- 코드 실습

Data Structures

- 데이터 구조
  - 데이터를 일정한 기준에 의해 모아 놓은 것
    (collection of data elements, structured in some way)
  - (종류) Built-in + 확장
  - Container type의 데이터구조
    - Sequence = mapping by element position (위치값 즉, index)
    - Mapping = mapping by element name (즉, key)

- Built-in Sequence
  - List
  - Tuples
  - String
  - Buffer objects
  - Xrange objects

- Sequence에 대한 주요 작업
  - Indexing
  - Slicing
  - Adding
  - Multiplying
  - Membership 확인

List

- 개념
- 특징:
  - ***"Mutable!"***
- Indexing
- Slicing
- Adding, Multiplying, Membership
- List관련 함수
- List methods
  - Method = object에 관련된 함수
    (a function tightly coupled to some object)
  - object.method(arguments)

- List에 대한 작업
  - 개요
    - (Sequence 공통)
    - Indexing, Slicing, Concatenating, multiplying
    - 여기에 추가해서 …
  - 변경작업: Item Assignment

```
>>> x =[1,1,1]
>>> x[1]
1
>>> x[1] =2
>>> x
[1, 2, 1]
```

- Slicing
  - 일정 범위 내의 element를 access하는 것
  - ☞주의 [index1, index2]
    – index1은 inclusive, index2는 exclusive

```
>>> tag = '<a href="http://www.python.org">Python web site </a>'
>>> tag[9:30]
'http://www.python.org'
>>> tag[32:-4]
```

```
url = input('Please enter the URL: ')
domain = url[:-4]

print("Domain name: " + domain)
```

- Sequence의 추가
  - 단, 같은 type일 것
- Membership

```
>>> permission = 'rw'
>>> 'w' in permission
True
>>> 'x' in permission
False

# spam filter에 이용
>>> subject = '$$$ Get rich now!! $$$'
>>> '$$$' in subject
True

>>> users = ['hky','foo','bar']
>>> input('Enter your name: ') in users
Enter your name: hky
True
```

- Multiplication
- Length, Minimum, Maximum
- List() 함수

```
>>> list('Hello')
['H', 'e', 'l', 'l', 'o']
>>> a = list('Hello')
>>> a
['H', 'e', 'l', 'l', 'o']
>>> ''.join(a)
'Hello'
```

List Methods

- append()
  - >>> lst = [1,2,3] # 변수명에 유의할 것
- count()
- extend()
  - # 주의: concatenation과 다름 - extended sequence is modified,
  - # vs. ordinary concatenation: 완전히 새로운 sequence가 반환됨
- index()
  - 처음 발견되는 항목의 index
- insert()
- pop()

- remove()
  - (주의1) 맨 첫 번째 항목부터 삭제됨
  - (주의2) "In-place change"이므로 되돌이킬 수 없음.
- reverse()
- sort()
  - ; sort lists 'in place'
  - >>> # 정렬된 copy본을 이용하고자 할 때 주의!!
  - # 즉, sort는 x를 변경시키지만 반환값은 없다 (nothing)
  - >>> # 이를 위해서는...
  - sort()의 option
    - # 고급의 sorting:
    - https://wiki.python.org/moin/HowTo/Sorting

- # sorted()는 어떤 sequence에든 이용 가능, 단 항상 list를 반환함

List Comprehension

- 개념
  - **간편하면서도 강력!**

```
>>> [x*x for x in range(10)]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
>>> # 3으로 나누어지는 수만?
>>> [x*x for x in range(10) if x % 3==0]
[0, 9, 36, 81]
>>> [(x,y) for x in range(3) for y in range(2)]
[(0, 0), (0, 1), (1, 0), (1, 1), (2, 0), (2, 1)]
result=[]
```

Tuples

- 개념
  - Immutable sequence
  - 구문: , 또는 (, ,)
- Operation
- Methods

- ☞ 주요 활용처
  - Map에서의 key (cf. list는 key가 될 수 없음)
  - 일부 내장함수에서의 반환값으로 이용

- 기본 사용방법

```
>>> 1,2,3
(1, 2, 3)
>>> (1,2,3)
(1, 2, 3)
>>> ()
()
>>> 42
42
>>> 42,
(42,)
>>> (42,)
(42,)
```

- tuple() 함수

Dictionary

- 개념
  - 내장된 mapping 데이터 타입 (KVP)
  - 사용처
- 생성방법
- dict 함수
- 기본적인 작업
  - len(d)
  - d[k]
  - d[k] = v
  - del d[k]
  - k in d

```
>>> eng2sp = { }
>>> eng2sp['one'] = 'uno'
>>> eng2sp['two'] = 'dos'
>>> print(eng2sp)
>>> print(eng2sp['two'])
```

- Dictionary Methods
  - clear
  - copy
  - fromkeys
  - get

- has_key
  - == k in d 단, Python 3에서 없어짐

```
>>> # d.has_key(k)
>>> d = {}
>>> d.has_key('name')
```

- items와 iteritems
- keys와 iterkeys
- setdefault
- update

- pop
- popitem
- values와 itervalues

보충 (1) pass, del, exec, eval

- pass() ; do nothing
  - **if name=='park':**
  - ** print('welcome')**
  - **elif name=='bush':**
  - ** pass**
  - **else:**
  - ** print('we are waiting')**
- del(): delete

- exec()
  - Execute a series of Python statements
  - …
- eval()
  - Evaluates a Python expression and return the resulting value

보충 (2) Sequence unpacking, …

- Sequence unpacking
- Chained assignment와 augmented assignment
- assert
  - 위험요소를 미리 드러나게 함
  - 일종의 checkpoint로 활용

```
x = y = somefunction()

>>> x=2
>>> x+=3
>>> x*=5
>>> x
25
>>>
>>> sname = 'foo'
>>> sname +='bar'
>>> sname *=3
>>> sname
'foobarfoobarfoobar'
```

String

- 개요

- 기초
  - "~~" '~~' """ ~~""" '~~~'
  - concatenation
  - input()
- Raw string
- str()과 repr()
  - str() ;human-readable,
  - repr() ; representations to be read by the interpreter
  - 같은 값이 반환될 때도 있음. 예: numbers, lists, dictionaries
  - 그러나, Strings, floating point numbers 등은 확연히 달라짐
- Formatting ; … 단, **string module**
- Methods

- Raw string
- str()과 repr()

```
>>> x=10 * 3.25
>>> y=200*200
>>> s = 'The value of x is ' + repr(x)
+ ', and y is ' + repr(y) + '...'
>>> print(s)
The value of x is 32.5, and y is 40000...
>>> # The repr() of a string adds
string quotes and backslashes:
>>> hello = 'Hello, world₩n'
>>> hellos = repr(hello)
>>> hellos
"'Hello, world₩₩n'"
>>> print(hellos)
'Hello, world₩n'
>>> repr((x,y, ('ham','eggs')))
"(32.5, 40000, ('ham', 'eggs'))"
```

- Formatted printing

String Methods

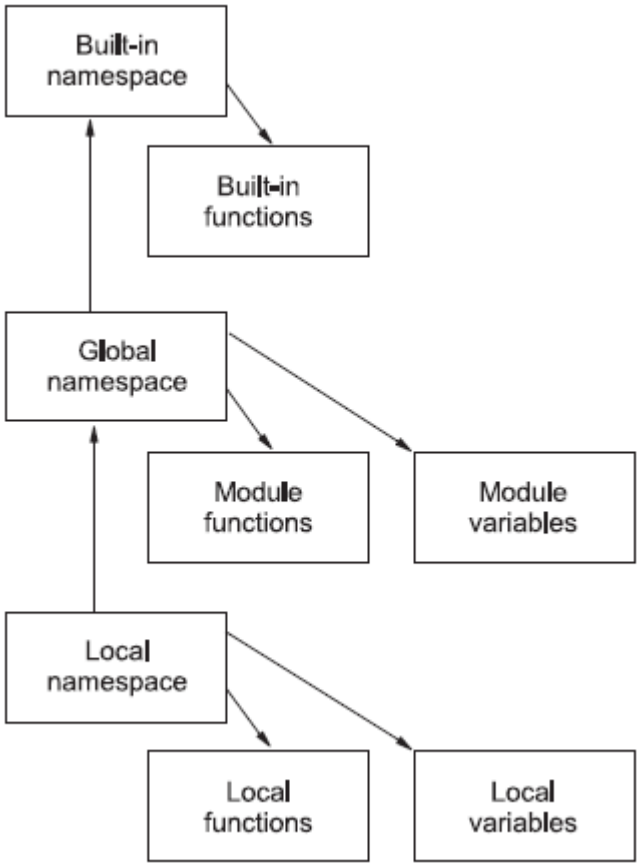
- join
- replace
- Split
- Strip
- find

Module

- 개념
  - 프로그램 파일 (Python, C, C++, ...)
  - 역할:
- Module 작성과 이용
  - Module작성
  - import ans
  - 검색경로
  - Private name
- Scoping Rule

- 개념
  - 프로그램 파일 (Python, C, C++, …)
  - 역할:
    - …
    - namespace를 통해 name clash를 방지
      - Namespace = 일종의 dictionary of identifiers
- Module 작성과 이용
  - Module작성
  - import 문
    - import 후 qualification이 필요함
  - 검색경로
    - 유의할 것
    - Python 경로 안에 설치, sys.path 변경, PYTHONPATH 이용, .pth 파일 작성
  - Private name

- Scoping Rule
  - 순서:
  - Local > Global > Built-in
    - locals()
    - globals()
    - dir(__builtins__)



- # module 작성 및 이용

```
>>> import II02_mymath
>>> area(2)
12.56636
>>> II02_mymath.pi
3.14159
>>> II02_mymath.area(5)
78.53975

>>> from II02_mymath import area
>>> area(10)
314.159
```

- # module 탐색경로

```
>>> import sys
>>> sys.path
['C:/Python31/DBnet',
'C:₩₩Python31₩₩Lib₩₩idlelib',
'C:₩₩Windows₩₩system32₩₩python31.zip',
'C:₩₩Python31₩₩DLLs',
'C:₩₩Python31₩₩lib',
'C:₩₩Python31₩₩lib₩₩plat-win',
'C:₩₩Python31',
'C:₩₩Python31₩₩lib₩₩site-packages']
```

II. 한글 텍스트 처리 (Text Processing)

- 개요
- 형태소 분석
- nltk

개요

- 텍스트 처리와 자연어 처리
  - 자연어를 입력 받아 이를 분석
  - 단계
    - 형태소분석,
    - 구문분석,
    - 의미분석

- 자연언어 분석
  - 형태소 분석 (lexical analysis, morphological analysis)
  - 구문 분석 (syntactic analysis, parsing)
  - 의미 분석 (semantic analysis)
  - 담화 분석 (discourse analysis)
  - 중의성 해소 (ambiguity resolution)

형태소 분석

- 개념
  - Morphological analysis
  - '단어(word)'(한국어 - 어절)를 구성하는 각 형태소를 인식하고 불규칙 활용이나 축약, 탈락 현상이 일어난 형태소는 원형을 복원하는 과정으로 기술.
  - 형태소(morpheme)란?
    - 1. 의미가 있는 최소한의 단위(minimally meaningful unit)
    - 2. 문법적, 관계적인 뜻을 나타내는 단어 또는 단어의 부분.
- 한국어에서의 형태소 분석
  - 한국어는 교착어로 문장 속에서 활용할 때 단어(어간)에 조사나 어미(어미)를 붙여 상황에 맞게 사용.
  - 이에 따라
    - 형태소의 품사에 대한 중의성 (+ 띄어쓰기 문제)
    - 분석기준을 '자모'로 할지 '음절'로 할지의 문제 등
    - 한자 및 다국어의 문제, 복합어의 문제
    - 한글 코드의 문제

- 어휘 분석(Lexical Analysis)
  - 목적
    - 단어/어절의 형태론적 특성에 대한 일반적인 규칙을 생성. 이를 통해 단어를 구성하는 형태소의 후보 목록을 작성.
    - (한국어) 어휘 형태소(=실질 형태소)와 문법 형태소(= 형식 형태소)가 결합되어 단어를 이루므로 단어로부터 각 형태소를 분리하고 분리된 형태소의 원형을 복원해야 함.
  - 형태소 분석기의 품사 태깅 (POS, Part of Speech)
    - Markov Model Tagger
    - Hidden Markov Model Tagger
    - Transformation-based Tagger
    - Neural network
    - Decision Tree …

분류 기준	형태소 분석유형
형태소 분석 모델	언어 독립적, 언어 종속적 모델
분석 알고리즘	규칙 기반, 사전 기반, 말뭉치 기반
분석 방향	Bottom-up parallel / Top-down predictive
어절 검색 방향	좌우 분석, 우좌 분석, 양방향 분석
결합 제약	어절형성 규칙과 결합제약 규칙, 접속 정보에 의한 결합 제약
문법형태소 사전	단위 형태소 수록, 결합 형태소 수록
형태소 처리 단위	자소 단위, 음절 단위

형태소 분석을 위한 품사 체계

체언	명사	단일명사/ 복합명사/ 외래어/ 미등록명사
	의존명사	일반의존명사/ 단위의존명사
	대명사	인칭대명사 /지시대명사
	수사	수사/ 아라비아 숫자
용언	동사	자동사 / 타동사
	형용사	일반형용사/ 존재형용사
	보조용언	보조동사/ 보조형용사
수식언 / 독립언	관형사	지시관형서/ 수관형사/ 성상관형사
	부사	일반부사/ 접속부사
	감탄사	감탄사
접사	접두사	접두사
	체언접미사	체언접미사
	용언화접미사	용언화접미사
조사	조사	격조사/ 보조사/ 접속조사
어미	어말어미	종결어미/ 연결어미/ 전성어미
	선어말어미	선어말어미

강승식, 한국어 형태소 분석과 정보검색, 2003년

오픈소스 형태소 분석기

- 목록
  - http://sjc333.egloos.com/2837613
  - http://kkma.snu.ac.kr/
  - https://kldp.org/node/93512
  - http://nlp.kookmin.ac.kr/HAM/kor/ham-intr.html
  - http://konlpy.org/ko/v0.4.3/
  - (보충 필요)

- 말뭉치 파일
  - http://ithub.korean.go.kr/user/main.do

konlpy

- 개요
  - 한국어 정보처리를 위한 파이썬 패키지
  - http://konlpy.org/ko/v0.4.4/
  - https://github.com/konlpy/konlpy
- Konlpy가 이용하는 형태소 분석 라이브러리
  - Kkma - http://kkma.snu.ac.kr/
  - Hannanum - http://semanticweb.kaist.ac.kr/hannanum/
  - Twitter - https://github.com/twitter/twitter-korean-text/
  - Komoran - http://www.shineware.co.kr/?page_id=835
  - Mecab - https://bitbucket.org/eunjeon/mecab-ko-dic
- 제공 기능
  - 한국어 corpus
  - 한국어 처리 유틸리티
  - 형태소 분석 및 품사 태깅
- 참고
  - http://kkma.snu.ac.kr/

nltk

- 개요
  - Python package that implements many standard NLP data structures, algorithms
  - First developed in 2001 as part of a CL course at University of Pennsylvania
- 오픈소스
  - http://www.nltk.org

주요 task	NLTK 모듈	기능
Accessing corpora	nltk.corpus	corpora 및lexicons에 대한 표준 인터페이스
String processing	nltk.tokenize	Sentence 및 word tokenizer
	nltk.stem	Stemmers
Part-of-speech tagging	nltk.tag	각종 품사 (POS) tagger
Classification	nltk.classify	Decision tree, maximum Entropy
	nltk.cluster	K-means
Chunking	nltk.chunk	Regular expressions, named entity tagging

텍스트 처리 framework의 문제

- 상용
  - …
- 오픈소스
  - Solr/Lucene
  - ELK (ElasticSearch/Logstash/Kibana) 등
  - …

기타

- API 활용
- 패키지
- 빅데이터

Python을 이용한 API 이용

- API란?
  - RESTful API (REpresentational State Transfer)
    - = producer와 consume간의 lighter weight 통신 → HTML 페이지의 XML 타입 정보를 읽어 들임
    - JSON 형태로 데이터를 제공 (그 밖에 XML, RSS 등)
  - 통상 인터넷의 HTTP (Hypertext Transfer Protocol)를 이용

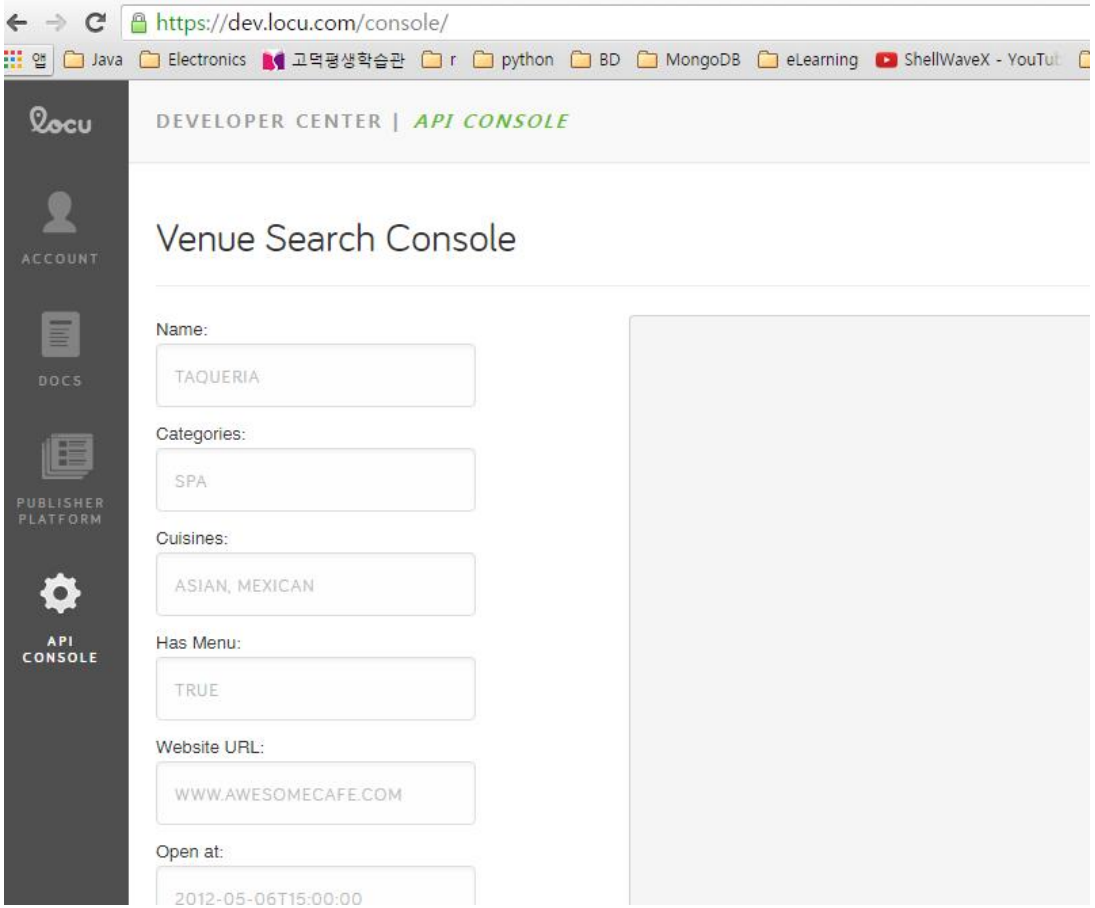
- REST 구성요소
  - Resource
    - 유일한 ID를 가지는 서버의 Resource를 대상으로 클라이언트는 요청을 보냄.
    - URI 의 예: '/groups/101/member/12532'
  - Method
    - Resource를 조작하는 동사.
    - 주요 method: GET, POST, PUT, DELETE 등

Method, CRUD, SQL 비교

Method	CRUD	SQL
DELETE	Delete	DELETE
POST	Create	INSERT
GET	Read	SELECT
PUT	Update	UPDATE

Open API 실습

- https://locu.com/



데이터분석 용 주요 패키지

- ipython,
- numpy,
- matplotlib,
- scipy
- pandas

빅데이터 처리

- Hadoop streaming의 이용
- 기타
  - …

map.py

```
#!/usr/bin/env python

import sys
from nltk.tokenize import wordpunct_tokenize

def read_input(file):
    for line in file:
        # split the line into tokens
        yield wordpunct_tokenize(line)

def main(separator='\t'):
    # input comes from STDIN (standard input)
    data = read_input(sys.stdin)
    for tokens in data:
        # what we output here will be the input for Reduce step,
        # i.e. the input for reducer.py
        #
        # tab-delimited; the trivial token count is 1
        for token in tokens:
            print '%s%s%d' % (word, separator, 1)

if __name__ == "__main__":
    main()
```

reduce.py

```
#!/usr/bin/env python

from itertools import groupby
from operator import itemgetter
import sys

def read_mapper_output(file, separator='₩t'):
    for line in file:
        yield line.rstrip().split(separator, 1)

def main(separator='₩t'):
    data = read_mapper_output(sys.stdin, separator=separator)
    for current_word, group in groupby(data, itemgetter(0)):
        try:
            total_count = sum(int(count) for current_word, count in group)
            print "%s%s%d" % (current_word, separator, total_count)
        except ValueError:
            pass

if __name__ == "__main__":
    main()
```

맺음말

분석 고도화

- 처리 방법론
  - Technology
    - DBMS
    - 검색시스템/framework
    - MapReduce
  - Raw 데이터 활용
    - 사전 – 다국어 처리, Thesaurus의 활용, n-gram – 복합어 처리
  - 처리 플랫폼
    - Linux – 가상머신 활용
- ML 활용 분석 기법
  - VSM (vector space model)
  - Text classification – Naïve Bayes, SVM
  - Clustering – Hierarchical clustering, Flat clustering
  - LSI (latent semantic indexing)
  - Link 분석
  - …