Python for Unstructured Data Analysis

2018.10.15 Hanyang University

Keywords

> **- 숫자형과 문자열 다루기**
>
> **- 리스트(list), 튜플(tuple), 딕셔너리(dictionary), 집합(set), 어레이(array), 그리고 자료형 간 변환**
>
> **- 비교, 조건, 불(booleans)**
>
> **- 파일 읽고 쓰기 : 피클(pickle), Text데이터 읽고 쓰기**
>
> **- Graph 자료구조**
>
> **- Graph 자료구조 기반 기초 알고리즘**

Jupyter Notebook에서 Markdown 다루기

1. 제목 달기 / 밑줄 긋기 / 들여쓰기

Heading 1

Heading 2

> ### Heading 3

2. LaTex(수식 쓰기)

$$e^x = \sum_{i=0}^{\infty} \frac{1}{i!} x^i$$

$e^{i\pi} + 1 = 0$

$$\mathbf{V}_1 \times \mathbf{V}_2 = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ \frac{\partial X}{\partial u} & \frac{\partial Y}{\partial u} & 0 \\ \frac{\partial X}{\partial v} & \frac{\partial Y}{\partial v} & 0 \end{vmatrix}$$

3. 표 만들기

	I	play	tennis
"I play tennis."	1	1	1
"I play."	1	1	0
"I, I play."	1	1	0

- markdown 표만들기를 도와주는 도구 : http://www.tablesgenerator.com/markdown_tables (http://www.tablesgenerator.com/markdown_tables)

4. 줄나누기 / 강조하기

The apparent complexity of its behavior over time is largely a reflection of the complexity of the environment in which it finds itself.

.....

At the level of cells or molecules ~~ants are demonstrably complex,~~
but these microscopic details of the inner environment may be largely irrelevant to the ant's behavior in relation to the outer environment.

5. 프로그램 코드 예쁘게 집어 넣기

- 파이썬 문법

```
def f(x):
  return x**2
```

- C++ 문법

```
if (i=0; i<n; i++) {
printf("hello %d\n", i);
x += 4;
}
```

! 수식, 코딩, 출력물, 설명 문서를 한번에 다루면 효율이 높아집니다. !

1. 숫자형 자료와 연산

In [1]:

```
x = 14
y = 3

print("x+y :", x+y)
print("x-y :", x-y)
print("x*y :", x*y)
print("x/y :", x/y)
print("round(x/y, 2) :", round(x/y, 2) ,"=> 소숫점 2째자리 반올림")
print("x**y :", x**y ,"=> x의 y승")
print("x//y :", x//y ,"=> x/y의 정수부분")
print("x%y :", x%y ,"=> x/y의 나머지, 짝수/홀수 여부 확인 시 유용")
```

```
x+y : 17
x-y : 11
x*y : 42
x/y : 4.666666666666667
round(x/y, 2) : 4.67 => 소숫점 2째자리 반올림
x**y : 2744 => x의 y승
x//y : 4 => x/y의 정수부분
x%y : 2 => x/y의 나머지, 짝수/홀수 여부 확인 시 유용
```

In [2]:

```
# 파이썬 특유의 연산 표시 : 다른 사람들의 코드를 읽어야할 경우가 있습니다.
x = 14
print("x : " ,x)
x += 1    # x에 1을 더합니다. 순차적으로 1씩 더할때 이용합니다.
print("x += 1 :" ,x)
x -= 4    # x에서 4를 뺍니다.
print("x -= 4 : " ,x)
x *= -2.6 # x에 -2.6을 곱합니다.
print("x *= -2.6 :",x)
x /= 5*y  # x를 y에 5를 곱한 숫자로 나눕니다.
print("x /= 5*y:" , x)
x //= 3.4 # x를 3.4로 나눈 후 반올림합니다.
print("x //= 3.4:" ,x)
```

```
x :  14
x += 1 : 15
x -= 4 :  11
x *= -2.6 : -28.6
x /= 5*y: -1.9066666666666667
x //= 3.4: -1.0
```

In [3]:

```
# 숫자 연산을 사용자의 입력을 받아 실행해 봅시다.

# 높이 (h)에서 공을 떨어뜨렸을 때 시간 (t) ms가 걸렸습니다. 공의 지름은 얼마일까요?

G = 9.81 # 지구에서 중력가속도, 상수는 따로 빼내어 할당하는 습관.
h = float(input("Enter the height of the tower: "))
t = float(input("Enter the time interval: "))
s = (G*t**2)/2
print("The height of the ball is", h-s, "meters")
```

```
Enter the height of the tower: 100
Enter the time interval: 1
The height of the ball is 95.095 meters
```

- 연습문제
  x = 14, y = 3 일때, 파이썬으로 x/y의 나머지를 %연산자를 사용하지 않고 구해보시오.

2. 문자형 자료와 연산

1) 인코딩(Encoding) & 디코딩(Decoding)

> **어렵고 또 어렵지만 Text Mining을 위한 데이터 전 처리시 겪어야만 하는 고난**

유니코드(Unicode) 인코딩:

> **- 컴퓨터는 결국 0,1 이라는 값만 인식할 수 있음.**
>
> **- 컴퓨터가 문자열을 처리할 수 있도록 문자를 숫자로 매핑, 표준화 시도(아스키:ASCII)**
>
> **- ASCII는 127개의 문자만 처리할 수 있는데, 이는 영어와 같은 알파벳만 처리 가능**
>
> **- 유니코드는 모든 언어를 표현할 수 있도록 넉넉하게 매핑 설계, 특히 문자열을 표현하는데 2~3byte가 필요한 한글 표현.(영어 알파벳은 1개 문자열에 1byte)**
>
> **- 유니코드 매핑 규칙 아래 여러 변종이 생겼는데 utf-8이 대표적. 최근 개발은 utf-8을 표준으로 하는 경우가 많음.**
>
> **- Window 기본 인코딩이 cp949로 되는 경우가 많으니 주의 필요.**

인코딩	특징	위키디피아
EUC-KR	한글완성형	https://ko.wikipedia.org/wiki/EUC-KR (https://ko.wikipedia.org/wiki/EUC-KR)
CP949	한글완성형(MS사)	https://ko.wikipedia.org/wiki/CP949 (https://ko.wikipedia.org/wiki/CP949)
UTF-8	유니코드8비트(한글1글자에 3바이트)	https://ko.wikipedia.org/wiki/UTF-8 (https://ko.wikipedia.org/wiki/UTF-8)

In [4]:

```
a = "노벨경제학상 수상자 폴 로머(paul Romer)는 차터시티(charter city)를 제안했다."
print(type(a))

# 인코딩 : 컴퓨터가 처리할 수 있는 형태로 변환
u8 = a.encode('utf-8')
print(type(u8))
print(u8)

print('-------------------------------------')

cp = a.encode('cp949')
print(type(cp))
print(cp)

print('-------------------------------------')

obt = "paul".encode('ascii')
print(type(obt))
print(obt)
```

```
<class 'str'>
<class 'bytes'>
b'\xeb\x85\xb8\xeb\xb2\xa8\xea\xb2\xbd\xec\xa0\x9c\xed\x95\x99\xec\x83
\x81 \xec\x88\x98\xec\x83\x81\xec\x9e\x90 \xed\x8f\xb4 \xeb\xa1\x9c\xe
b\xa8\xb8(paul Romer)\xeb\x8a\x94 \xec\xb0\xa8\xed\x84\xb0\xec\x8b\x9c
\xed\x8b\xb0(charter city)\xeb\xa5\xbc \xec\xa0\x9c\xec\x95\x88\xed\x9
6\x88\xeb\x8b\xa4.'
-------------------------------------
<class 'bytes'>
b'\xb3\xeb\xba\xa7\xb0\xe6\xc1\xa6\xc7\xd0\xbb\xf3 \xbc\xf6\xbb\xf3\xc
0\xda \xc6\xfa \xb7\xce\xb8\xd3(paul Romer)\xb4\xc2 \xc2\xf7\xc5\xcd\x
bd\xc3\xc6\xbc(charter city)\xb8\xa6 \xc1\xa6\xbe\xc8\xc7\xdf\xb4\xd
9.'
-------------------------------------
<class 'bytes'>
b'paul'
```

In [5]:

```
# 디코딩 : 사람이 읽을 수 있는 형태로 변환
du8 = u8.decode('utf-8')
print(type(du8))
print(du8)

print('-------------------------------------')

dcp = u8.decode('cp949')
print(type(dcp))
print(dcp)
```

```
<class 'str'>
노벨경제학상 수상자 폴 로머(paul Romer)는 차터시티(charter city)를 제안했다.
-------------------------------------

---------------------------------------------------------------------------
-----
UnicodeDecodeError                        Traceback (most recent call
 last)
<ipython-input-5-f65f57a61743> in <module>()
      6 print('-------------------------------------')
      7
----> 8 dcp = u8.decode('cp949')
      9 print(type(dcp))
     10 print(dcp)

UnicodeDecodeError: 'cp949' codec can't decode byte 0xeb in position
 0: illegal multibyte sequence
```

In [6]:

```
# bytes로부터 인코딩 알아내기
import chardet
print(chardet.detect(cp))
print(chardet.detect(u8))
print(chardet.detect(obt))
```

```
{'language': 'Korean', 'encoding': 'EUC-KR', 'confidence': 0.99}
{'language': '', 'encoding': 'utf-8', 'confidence': 0.99}
{'language': '', 'encoding': 'ascii', 'confidence': 1.0}
```

인코딩 관련 문제를 겪지 않기 위한 프로세스

- 주고 받는 파일의 인코딩을 통일한다. (ex : UTF-8)

- 프로세스의 첫 단계에서 입력 받은 바이트 문자열을 유니코드 문자열로 디코딩한다.

- 디코딩된 문자열 상태에서 자료처리를 한다.

- 처리된 자료는 통일된 인코딩으로 파일로 저장한다.(ex : UTF-8)

1. utf-8로 작성된 파일 읽기

```
import codecs
with codecs.open('utf8.txt', encoding='utf8', mode = 'r') as f:
    data = f.read()  # 유니코드 문자열
```

2. unicode 문자열로 프로그램 수행하기

```
data = function(data)
```

3. utf-8로 수정된 문자열 저장하기

```
with open('func_utf8.txt', encoding='utf8', mode='w') as f:
    f.write(data)
```

- source code 인코딩

source code(*.py 파일)도 문자로 이루어진 파일임으로 인코딩 타입이 필요하다. 파이썬3.0부터는 기본 Default로 설정되어 있으나 다른 사람이 짠 옛 코드에는 다른 인코딩이 나타날 수 있다. source code 인코딩은 source파일 첫 줄에*
-- coding: utf-8 -*-
로 표시하여 지정한다.

2) 문자열 처리에 대해 알아둘만한 사항

문자열 안에 변수 넣기

In [7]:

```
print("1.정수형 넣기")
print("I ate %d apples." % 3)
print("2.문자열 넣기")
print("I ate %s apples." % "five")
print("3.정수형 변수 넣기")
number = 3
print("I eat %d apples." % number)
print("4.2개 이상의 변수 넣기")
number = 10
day = "three"
print("I ate %d apples. so I was sick for %s days." % (number, day))
print("4.소숫점 변수 넣기")
import math
print("The value of pi is %0.2f." % math.pi)
```

```
1.정수형 넣기
I ate 3 apples.
2.문자열 넣기
I ate five apples.
3.정수형 변수 넣기
I eat 3 apples.
4.2개 이상의 변수 넣기
I ate 10 apples. so I was sick for three days.
4.소숫점 변수 넣기
The value of pi is 3.14.
```

문자열 관련 함수

In [8]:

```
print("1.문자 갯수 세기")
print("infinity".count('in'))

print("2.문자열 양쪽 공백 지우기")
print("  infinity  ".strip())

print("3.문자열 바꾸기")
print("infinity".replace("inf","def"))

print("4.문자열 나누기")
print("infinity".split("ni"))

print("5.슬라이싱으로 문자열 나누기")
print("infinity"[0])
print("infinity"[:2])
print("infinity"[2:])
print("infinity"[3:5])
print("infinity"[-2:])
print("infinity"[:-2])
```

```
1.문자 갯수 세기
2
2.문자열 양쪽 공백 지우기
infinity
3.문자열 바꾸기
definity
4.문자열 나누기
['infi', 'ty']
5.슬라이싱으로 문자열 나누기
i
in
finity
in
ty
infini
```

3. 파이썬 자료구조 및 문법

1) Booleans & Assignment & Comparisons & Conditional expression

In [9]:

```
print(5 == 4)
print(4 == 4)
print(4 > 4)
print(4 < 4)
print(4 <= 4)
print(7 >= 9)
print(7 != 9)
print(type((7 != 9)))
```

```
False
True
False
False
True
False
True
<class 'bool'>
```

In [10]:

```
x = 5+4
y = 2*x
print(y)

print(2**(y+1/2) if x+10 < 0 else 2**(y-1/2))

if x+10 < 0 :
    print(2**(y+1/2))
else :
    print(2**(y-1/2))
```

```
18
185363.80004736633
185363.80004736633
```

2) 집합(Sets) : {1,2} 중복과 순서가 없다.

In [11]:

```
# 문자열의 중복을 제거하거나 unique한 단어의 수, 특정 단어의 포함여부 등을 확인할 때 사용
set1 = {1+2, 3, 4}
set2 = {3, 4, 1+2}
print(set1)
print(set2)
print(sum(set1))
print(len(set1))

S = {1,2,3}
print(2 in S)
print(7 in S)

#Union
U = set1 | S
print(U)

#Intersetion
I = set1 & S
print(I)

#변형하기
U.add(20)
print(U)
U.remove(3)
print(U)
U.update({40,50})
print(U)
```

```
{3, 4}
{3, 4}
7
2
True
False
{1, 2, 3, 4}
{3}
{1, 2, 3, 4, 20}
{1, 2, 4, 20}
{1, 2, 50, 4, 20, 40}
```

In [12]:

```
Z = U
print(Z)
Z.remove(4)
print(Z, U)
# U에서도 4가 삭제된다. Python은 1개의 copy data structure를 저장한다.
```

```
{1, 2, 50, 4, 20, 40}
{1, 2, 50, 20, 40} {1, 2, 50, 20, 40}
```

In [13]:

```
T = U.copy()
T.add(90)
print(U, T)
# copy를 이용하면 T에는 90이 추가되었으나 U에는 추가되지 않았다.
```

```
{1, 2, 50, 20, 40} {1, 2, 40, 50, 20, 90}
```

In [14]:

```
comp_set1 = {2*x for x in {1,2,3}}
print(comp_set1)

comp_set2 = {x*x for x in {1,2} | {1,2,3}  if x > 2}
print(comp_set2)

comp_set3 = {x*y for x in {1,2} for y in {1,2,3} if x!=y}
print(comp_set3)

empty_set4 = {x*x for x in {1,2} | {1,2,3}  if x > 4}
print(empty_set4)

error_set_in_set = {{1,3}, 3}
```

```
{2, 4, 6}
{9}
{2, 3, 6}
set()

---------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
 last)
<ipython-input-14-410c06ec8fff> in <module>()
     11 print(empty_set4)
     12
---> 13 error_set_in_set = {{1,3}, 3}

TypeError: unhashable type: 'set'
```

In [15]:

```
{1,2,3}[0]
```

```
---------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
 last)
<ipython-input-15-71bf5e044980> in <module>()
----> 1 {1,2,3}[0]

TypeError: 'set' object does not support indexing
```

3) 리스트(list) : [1,2] 가장 많이 쓰이는 자료구조

In [16]:

```
a = list()
print(a)

b = [[1,1+1,4-2], {2*2, 5, 6}, "python"]
print(b)

c = [1,2,3] + ["my word"]
print(c)

c.append(b)
print(c)
```

```
[]
[[1, 2, 2], {4, 5, 6}, 'python']
[1, 2, 3, 'my word']
[1, 2, 3, 'my word', [[1, 2, 2], {4, 5, 6}, 'python']]
```

In [17]:

```
# list comprehensions
print([2*x for x in {2,1,3,4,5}])
print([2*x for x in [2,1,3,4,5]])
print([x*y for x in [2,1,3,4,5] for y in [10,20,30]])
print([[x,y] for x in ['A','B','C'] for y in [1,2,3]])
```

```
[2, 4, 6, 8, 10]
[4, 2, 6, 8, 10]
[20, 40, 60, 10, 20, 30, 30, 60, 90, 40, 80, 120, 50, 100, 150]
[['A', 1], ['A', 2], ['A', 3], ['B', 1], ['B', 2], ['B', 3], ['C', 1],
['C', 2], ['C', 3]]
```

In [18]:

```
L = [x*10 for x in range(1,9)]
print(L)
print(L[5:]) # 위치 5부터 그 다음
print(L[:5]) # 위치 5부터 마지막까지 제외
print(L[3:5]) # 위치 3부터 위치 5부터 마지막까지 제외
print(L[::2]) # 처음부터 매 2번째 마다
print(L[1::2])# 위치 1에서부터 매 2번째 마다
print(L[:-2]) # 마지막에서 출발하여 위치 2
print(L[-2:]) # 마지막 2에서 출발하여 나머지
print(L[::-1]) # 마지막에서부터 전체
```

```
[10, 20, 30, 40, 50, 60, 70, 80]
[60, 70, 80]
[10, 20, 30, 40, 50]
[40, 50]
[10, 30, 50, 70]
[20, 40, 60, 80]
[10, 20, 30, 40, 50, 60]
[70, 80]
[80, 70, 60, 50, 40, 30, 20, 10]
```

In [19]:

```
list_of_list = [[1,1],[2,4],[3,9]]
flat_list = [x for sublist in list_of_list for x in sublist]
print(flat_list)

unpack_list = [y for [x,y] in list_of_list]
print(unpack_list)

import numpy as np
np_array = np.array(list_of_list)
print(np_array)
```

```
[1, 1, 2, 4, 3, 9]
[1, 4, 9]
[[1 1]
 [2 4]
 [3 9]]
```

4) 튜플(tuple) : (1,2) 원소의 삭제, 변경이 안된다.

In [20]:

```
my_tuple = ("A","B","C")
print(my_tuple[0])

unpack_tuple = [y for (x,y) in [(1,'A'),(2,'B'),(3,'C')]]
print(unpack_tuple)

print(list(zip([1,2,3],[4,5,6])))
characters = ['Neo', 'Morpheus', 'Trinity']
actors = ['Keanu', 'Laurence', 'Carrie-Anne']
print(set(zip(characters, actors)))
```

```
A
['A', 'B', 'C']
[(1, 4), (2, 5), (3, 6)]
{('Trinity', 'Carrie-Anne'), ('Morpheus', 'Laurence'), ('Neo', 'Kean
u')}
```

In [21]:

```
# tuple comprehesions is NOT a tuple, IT's a generator
print((i for i in [1,2,3]))
print([i for i in [1,2,3]])
print(list(i for i in [1,2,3]))
```

```
<generator object <genexpr> at 0x108638938>
[1, 2, 3]
[1, 2, 3]
```

5) Python Comprehension :

iterable한 오브젝트를 생성하기 위한 방법 중 하나

-List comprehension

-Set comprehension

-Dictionary comprehension

-Generator Expression :
한번에 모든 원소를 반환하지 않고, 한번에 하나의 원소만 반환하는 generator를 생성한다.

https://mingrammer.com/introduce-comprehension-of-python/ (https://mingrammer.com/introduce-comprehension-of-python/)

6) 딕셔너리(Dictionary) : {Key1:Value1, Key2:Value2 ...}

In [22]:

```
# dictionary는 Key로 indexing 한다.
# Key는 중복이 없어야 한다.

my_dict = {'Neo' : 40000, 'Thanos' : 100000}
print(my_dict['Neo'])

my_dict['Iron Man'] = 200000
print(my_dict)
print(my_dict.keys())
print(my_dict.values())
print(my_dict.items())
print([x for x in my_dict.items()])
print([k + "'s power level : " + str(v) for (k,v) in my_dict.items()])
```

```
40000
{'Neo': 40000, 'Thanos': 100000, 'Iron Man': 200000}
dict_keys(['Neo', 'Thanos', 'Iron Man'])
dict_values([40000, 100000, 200000])
dict_items([('Neo', 40000), ('Thanos', 100000), ('Iron Man', 200000)])
[('Neo', 40000), ('Thanos', 100000), ('Iron Man', 200000)]
["Neo's power level : 40000", "Thanos's power level : 100000", "Iron M
an's power level : 200000"]
```

In [23]:

```
key_list = ['A', 'B', 'C']
value_list = [1, 2, 3]
new_dict = dict(zip(key_list, value_list))
print(new_dict)
```

```
{'B': 2, 'A': 1, 'C': 3}
```

In [24]:

```
print([2*x for x in {4:'a', 3:'b'}.keys()])
print([x+'c' for x in {4:'a', 3:'b'}.values()])
print([k for k in {'a':1, 'b':2}.keys() | {'b':3, 'c':4}.keys()])
print([k for k in {'a':1, 'b':2}.keys() & {'b':3, 'c':4}.keys()])
```

```
[6, 8]
['bc', 'ac']
['a', 'c', 'b']
['b']
```

4. 파일 읽고 쓰기

1) 이쯤에서 메모리에 담겨 있는 변수들을 확인해 봅시다.

In [25]:

```
whos
```

```
Variable       Type       Data/Info
-----------------------------------
G              float      9.81
I              set        {3}
L              list       n=8
S              set        {1, 2, 3}
T              set        {1, 2, 40, 50, 20, 90}
U              set        {1, 2, 50, 20, 40}
Z              set        {1, 2, 50, 20, 40}
a              list       n=0
actors         list       n=3
b              list       n=3
c              list       n=5
characters     list       n=3
chardet        module     <module 'chardet' from '/<...>ges/chardet/__
init__.py'>
comp_set1      set        {2, 4, 6}
comp_set2      set        {9}
comp_set3      set        {2, 3, 6}
cp             bytes      b'\xb3\xeb\xba\xa7\xb0\xe<...>xbe\xc8\xc7\xd
f\xb4\xd9.'
day            str        three
du8            str        노벨경제학상 수상자 폴 로머(paul Rome<...>차터시티(ch
arter city)를 제안했다.
empty_set4     set        set()
flat_list      list       n=6
h              float      100.0
key_list       list       n=3
list_of_list   list       n=3
math           module     <module 'math' from '//an<...>3.5/lib-dynloa
d/math.so'>
my_dict        dict       n=3
my_tuple       tuple      n=3
new_dict       dict       n=3
np             module     <module 'numpy' from '//a<...>kages/numpy/__
init__.py'>
np_array       ndarray    3x2: 6 elems, type `int64`, 48 bytes
number         int        10
obt            bytes      b'paul'
s              float      4.905
set1           set        {3, 4}
set2           set        {3, 4}
t              float      1.0
u8             bytes      b'\xeb\x85\xb8\xeb\xb2\xa<...>xed\x96\x88\xe
b\x8b\xa4.'
unpack_list    list       n=3
unpack_tuple   list       n=3
value_list     list       n=3
x              int        9
y              int        18
```

In [26]:

```
del(G,I,s)
```

In [27]:

```
all = [var for var in globals() if var[0] != "_"]
for var in all:
    del globals()[var]
```

2) 텍스트 파일 읽고 쓰기

In [28]:

```
import os
import glob
import codecs

floc = '/Users/hanjudong/work/W2018_HYU_PYTHON'
econ_flist = glob.glob(floc+'/econ_nobel/*.txt')
phys_flist = glob.glob(floc+'/phys_nobel/*.txt')
```

In [29]:

```
def doctolist(flst, file, category):
    doclist = [category]
    with codecs.open(file, 'r' , encoding='utf-8', errors='ignore') as f:
        doclist.append(file.replace(".txt","").replace(floc,""))
        data = [f.read().lower()]
        doclist.append(data[0])
    return doclist
```

In [30]:

```
econ_txt = (
 [doctolist(econ_flist, file, 'econ_nobel') for file in econ_flist]
)

phys_txt = (
 [doctolist(econ_flist, file, 'phys_nobel') for file in econ_flist]
)
```

In [31]:

```
to_write = econ_txt[0][0]
print(to_write)
```

```
econ_nobel
```

In [36]:

```
with open(floc+'/test_write.txt', 'w') as t:
    t.write(to_write)
```

In [37]:

```
!cat /Users/hanjudong/work/W2018_HYU_PYTHON/test_write.txt
```

```
econ_nobel
```

In [38]:

```
# 추가 모드로 쓰기, with 를 빼면 close() 로 파일을 닫아줘야함.
a = open(floc+'/test_write.txt', 'a')
for i in range(1,3):
    if i == 1 :
        data = "\n%d번째 줄에 자료를 추가합니다. \n" % (i+1)
    else :
        data = "%d번째 줄에 자료를 추가합니다. \n" % (i+1)
    a.write(data)
a.close()
```

In [39]:

```
!cat /Users/hanjudong/work/W2018_HYU_PYTHON/test_write.txt
```

```
econ_nobel
2번째 줄에 자료를 추가합니다.
3번째 줄에 자료를 추가합니다.
```

참고

excel 파일을 읽고 쓰는 것은 xlrd, ,openpyxl, xlwt 등을 알아보고 활용해 봅시다.

http://www.python-excel.org/ (http://www.python-excel.org/)

3) Pickle : 파이썬 고유의 자료형을 그대로 저장하고 불러온다.(str로 처리하지 않고)

In [40]:

```
import pickle
list = ['a', 'b', 'c']
with open ('test_pickle.txt', 'wb') as f:
    pickle.dump(list, f)

with open ('test_pickle.txt', 'rb') as r:
    back_list = pickle.load(r)
print(back_list)
```

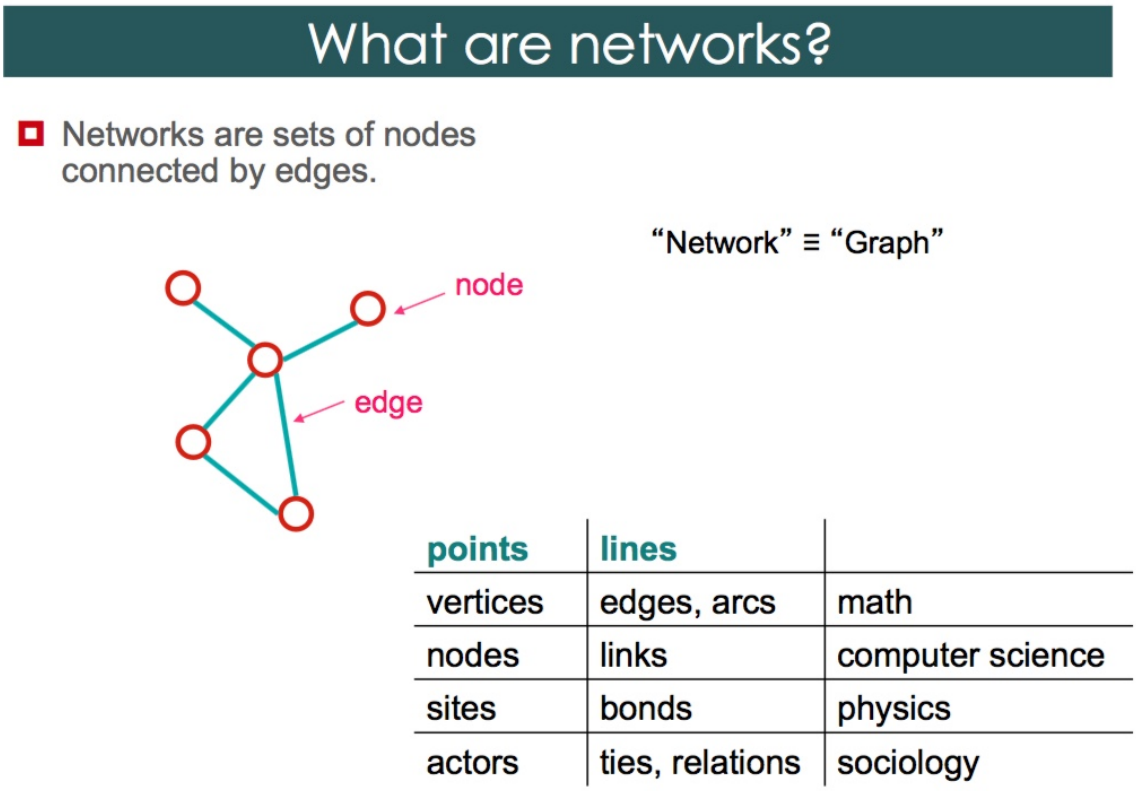
```
['a', 'b', 'c']
```

5. Graph

In [41]:

```
from IPython.display import Image
Image(filename='sn.jpeg')
```

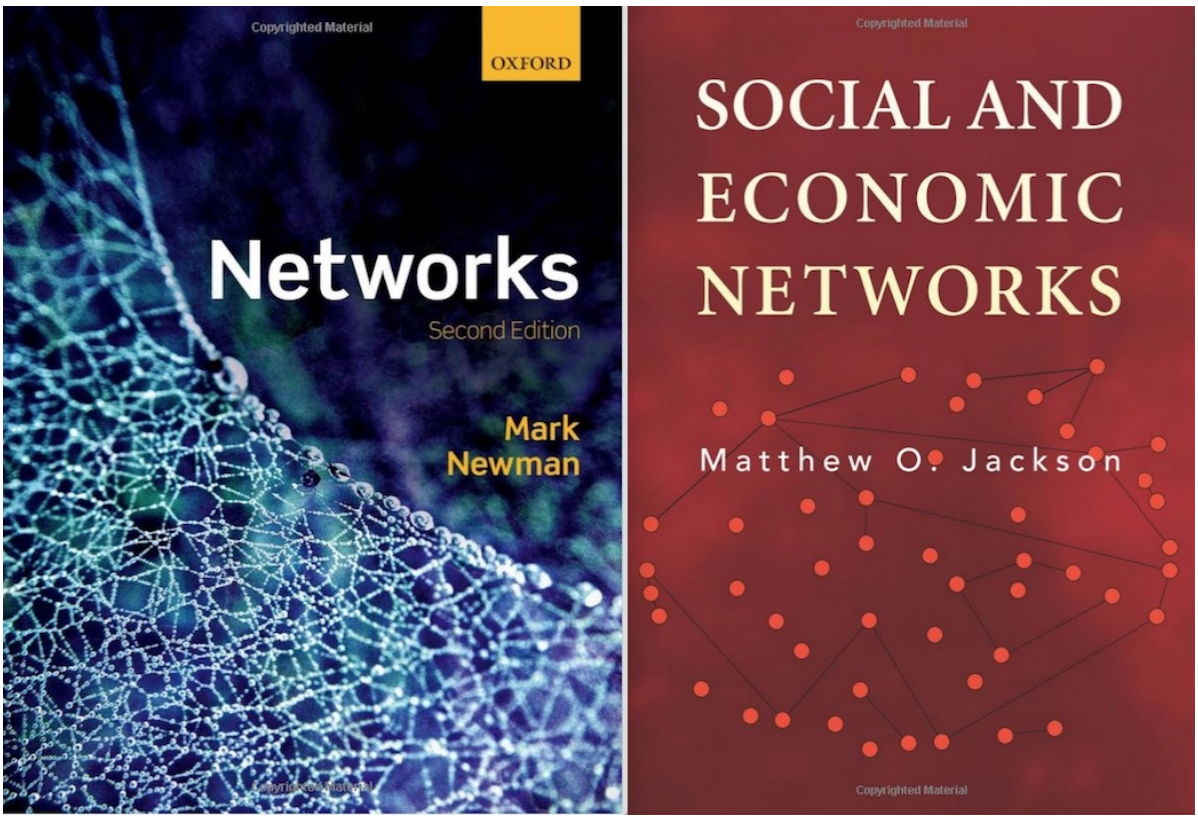
Out[41]:



In [42]:

```
Image(filename='Nbooks.jpeg')
```

Out[42]:



In [43]:

```
import networkx as nx
G = nx.read_gexf('LesMiseables.gexf')
```

In [44]:

```
G.nodes(data=True)
```

Out[44]:

```
[('42',
  {'Modularity Class': 7,
   'label': 'Anzelma',
   'viz': {'color': {'a': 1.0, 'b': 72, 'g': 81, 'r': 236},
    'position': {'x': 189.69513, 'y': -346.50662, 'z': 0.0},
    'size': 9.485714}}),
 ('7',
  {'Modularity Class': 0,
   'label': 'Cravatte',
   'viz': {'color': {'a': 1.0, 'b': 72, 'g': 81, 'r': 236},
    'position': {'x': -382.69568, 'y': 475.09113, 'z': 0.0},
    'size': 4.0}}),
 ('68',
  {'Modularity Class': 7,
   'label': 'Gueulemer',
   'viz': {'color': {'a': 1.0, 'b': 72, 'g': 81, 'r': 235},
    'position': {'x': 78.4799, 'y': -347.15146, 'z': 0.0},
    'size': 28.685715}}),
```

In [45]:

```
# 이름(Label)을 알때 Node ID를 찾기
for i in range(len(G.nodes())):
    if G.nodes(data=True)[i][1]['label'] == 'Valjean':
        print(G.nodes(data=True)[i][0])

# Node ID를 알때 이름(Label)을 찾기
for i in range(len(G.nodes())):
    if G.nodes(data=True)[i][0] == '11':
        print(G.nodes(data=True)[i][1]['label'])
```

```
11
Valjean
```

In [46]:

```
G.edges(data=True)[1]
```

Out[46]:

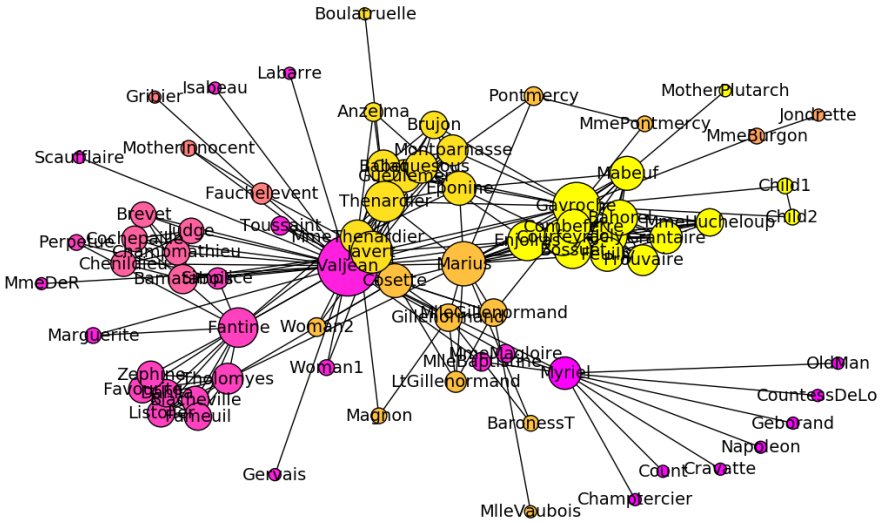
```
('42', '41', {'id': '99', 'weight': 2.0})
```

In [47]:

```
node_class = {G.nodes(data=True)[i][0] : int(G.nodes(data=True)[i][1]['Modularity Cl
node_label = {G.nodes(data=True)[i][0] : G.nodes(data=True)[i][1]['label'] for i in
node_size = {G.nodes(data=True)[i][0] : 25*G.nodes(data=True)[i][1]['viz']['size'] f
```

In [48]:

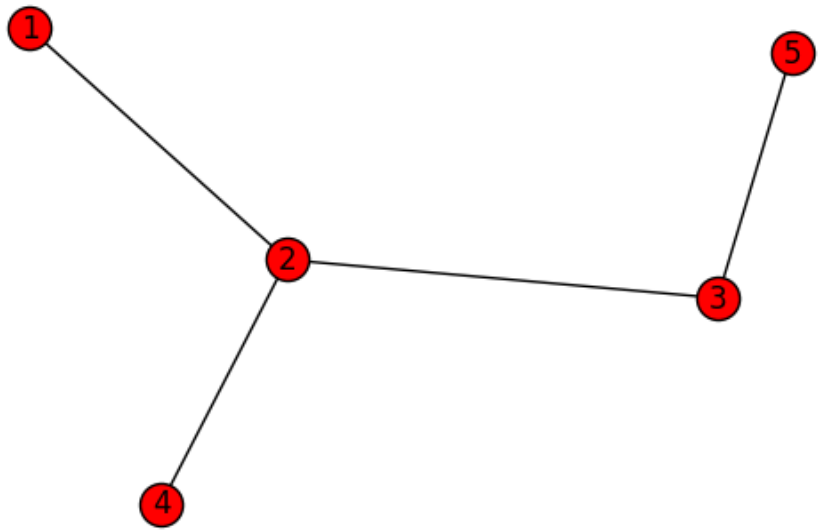
```
import matplotlib.pyplot as plt
plt.figure(figsize=(13,8), dpi=80)
values = [node_class.get(node) for node in G.nodes()]
size = [node_size.get(node) for node in G.nodes()]
nx.draw(G, cmap = plt.get_cmap('spring'), node_color = values, node_size = size ,lab
plt.show()
```



1) adjacency list

In [49]:

```
lines = ['1 2',
         '2 3 4',
         '3 5']
J = nx.parse_adjlist(lines, nodetype=int)
nx.draw(J, with_labels=True)
plt.show()
print(J.edges())
```




```
[(1, 2), (2, 3), (2, 4), (3, 5)]
```

2) Adjacency Matrix

In [50]:

```
A = nx.adjacency_matrix(J)
print(A.todense())
```

```
[[0 1 0 0 0]
 [1 0 1 1 0]
 [0 1 0 0 1]
 [0 1 0 0 0]
 [0 0 1 0 0]]
```

3) Centrality

> 어떠한 Node가 얼마나 상대적으로 중요한가?

1. Degree Centrality : the fraction of nodes it is connected to

2. Eigenvector Centrality : each nodes have a score proportional to the sum of the scores of its neighbors

- IDEA :

$$x_i' = \sum_j A_{ij} x_j$$

After t steps a vector of centralities $x(t)$ given by

$$x(t) = A^t x(0)$$

Write $x(0)$ as a linear combination of the eigenvectors v_i of the adjacency matrix A

$$x(0) = \sum_i c_i v_i$$

Choice some appropriate(normalization) constants c_i

$$x(t) = A^t \sum_i c_i v_i = \sum_i c_i k_i^t v_i = k_1^t \sum_i c_i (\frac{k_i}{k_1})^t v_i$$

where

k_i **: eigenvalues of** A

k_1 **: the largest of eigenvalue of** A

so, $\frac{k_i}{k_1} < 1$ for all $i \neq 1 \Rightarrow t \rightarrow \infty$, we get $x(t) \rightarrow c_1 k_1^t v_1$

$A \rightarrow k_1$ Centrality x satisfies

$$Ax = k_1 x$$

Eigenvector Centrality x_i

$$x_i = k_1^{-1} \sum_j A_{ij} x_j$$

Node 4는 degree centrality는 Node5와 같지만 Node 2와 직접 연결되어 있어 eigenvalue centrality는 더 크다.

In [51]:

```
dc = nx.degree_centrality(J)
print("degree_centrality :" , dc)
ec = nx.eigenvector_centrality_numpy(J)
print("eigenvector_centrality :" , ec)
```

```
degree_centrality : {1: 0.25, 2: 0.75, 3: 0.5, 4: 0.25, 5: 0.25}
eigenvector_centrality : {1: 0.35355339059327395, 2: 0.653281482438188
3, 3: 0.4999999999999999, 4: 0.35355339059327373, 5: 0.270598050073098
7}
```

5. 다루지 않았지만 알아두면 좋을 것들

- 데이터 처리 : 정규표현식(regular expression), 에디터(Vim, Sublime, Emacs, Pycharm)

- Graph : Breadth-Fisrt Search(너비우선탐색), Bepth-Fisrt Search(깊이우선탐색), Shortest Path(최단경로)

6. 연습

- 노벨 물리학상/경제학상 수상 연설(phys_nobel, econ_nobel 폴더) txt 파일을 파이썬으로 읽어 들여오고,

- pt_arrange.ipynb 파일을 참조하여

1) 품사별로 Tagging하고

2) 등장하는 전체 단어의 갯수를 세어본다.

3) 등장하는 전체 명사의 갯수를 세어본다.

4) 명사+명사 의 갯수를 세어본다.

In [ ]: