<영어음성학 정리 노트> 2017130844 최하빈

1. 영어의 consonants & vowels

-> grouping할 수 있어야 함.

ex 1) 유성음(voiced sound) : b, d, g, m, n, ŋ, v, ð, z, ʒ, l, w, r, j(y), dʒ, vowels
/무성음(voiceless sound) : p, t, k, f, θ, s, ʃ, h, tʃ

ex 2) 비음(nasal sound) : m / n/ ng **ex 3)** 이중모음(diphthongs) / 단모음(monophthongs)

2. Phonetics 음성학

-> '사람이 하는 말(speech)'에 대한 연구 / '물리적'인 부분에 초점을 둔다.
(반면, Phonology 음운론은 'sound system'에 대한 연구/'인지적'인 부분에 초점을 둔다)

-> Articulatory phonetics(조음): '말'을 할 때
Acoustic phonetics(음향): 말을 한 것이 '공기'를 타고 움직이는 물리적인 원리. 공기와 소리가 어떻게 작용하는가.
(사람이 개입되지 않은 완전한 '물리적'인 영역 <-> articulation은 사람의 입이 개입)
Auditory phonetics(청각): '귀'를 통해 어떻게 듣는가에 대한 사람이 수반된 mechanism
(성대 대신 '고막'의 움직임, '물리적'인 영역) (*귓바퀴: 소리를 증폭시키기 위함)

1) Articulation 조음

: speech를 만들 때 시작이 되는 mechanism
->'성대의 떨림 횟수'는 소리의 높낮이(pitch)와 관련 있음. / '입모양'(혀의 위치, 턱..)이 다른 소리를 발음할 수 있게 함.
('턱의 높낮이'가 변하지 않더라도 입을 이용해 다른 발음 가능 -> 주된 요인 아님)
: *한국어와 영어의 차이점
->'한국어'는 음절이 반복됨- '턱'을 위주로 쓰는 언어/ 반면, '영어'는 stress를 기본으로 반복됨- '혀'를 위주로 쓰는 언어

① Vocal tract

①-1. Upper vocal tract : lip, teeth, alveolar ridge, hard palate, soft palate(velum), uvula , pharynx, larynx -> 움직이지 않음! (fixed)
①-2. Lower vocal tract : lip, 'tongue'(tip/front/back/root/ blade/center), epiglottis -> 움직임!
(*epiglottis: 침, 음식이 기도로 가는 길을 막아줌. 식도로 갈 수 있도록)

② 5 speech organs (=constrictors =articulators)

②-1. Oro-nasal process in 'Velum'

: 'nasal sound'와 nasal sound가 아닌 것을 구분하는 과정.
: velum(soft palate)이 'lower'되어 nasal tract가 열리면 nasal sound('m/n/ŋ') <-> 'raised'되어 막히면 비음이 아닌 것.
*Q) velum이 raise가 되었다. nasal tract가 막혔을까요, 열렸을까요?? : 막혔다. (모든 모음, 비음 빼 모든 자음들)
***Q) 코로 숨을 쉴 때 velum이 raise될까 혹은 lower될까? (매년 시험문제)**
:숨을 쉴 때는 lower된 상태(코로 숨을 쉴 때 nasal tract가 열려야 함. '아-'와 같은 소리를 내는 순간 velum이 올라감.)

②-2. Phonation process in 'Larynx'

: (성대에서의 떨림 유무로)'voiced sound(유성음)'와 'voiceless sound(무성음)'를 구분하는 과정.
: larynx가 열려서 공기가 통과하면 voiceless sound(무성음) - 무성 자음
<-> larynx가 막혀서 기압으로 인해 진동이 생기면 voiced sound(유성음) - 모든 모음, 유성 자음

②-3. Articulatory process in 'lips/ tongue tip / tongue body'

: 'lips/ tongue tip(혀의 앞쪽)/ tongue body(혀의 뒷쪽)'는 주요 'constrictors'(constriction을 만드는 주체)
-> 각각의 constrictor(articulator)는 'constriction location(CD)'와 'constriction degree(CD)'에 의해 더 세분화됨.

②-3-ⓐ. Constriction Location(CL) : 위치 -> "앞 /뒤"
: Lips (Bilabial/ Labiodental), Tongue Tip (Palatal/ Velar), Tongue Body (Dental/ Alveolar/ Retroflex/ Palato-Alveolar)

②-3-ⓑ. Constriction Degree(CD) : 정도 -> "상/하" (각각의 constrictor가 upper part를 얼마나 세게 쳤는지)
: Stops , Fricatives , Approximants,(여기까진 다 Consonants) , Vowels (-> 뒤로 갈수록 덜 막힌 것)

*** 모든 음은 "Constrictors, CL, CD, Velum(높낮이), Larynx(open/close)"를 명시하여 구분 가능하다.**
(모든 '모음'은 constrictor로서 "tongue body"만 사용)

Q) Constrictor: tongue tip / CL: alveolar / CD: stops / Velum: raised / Larynx의 틈(glottis): open => 이 소리는 무슨 소리일까요? : /t/
Q) 모음과 같은 constrictor를 쓰는 것의 예를 드시오. : k , g , ng, j, vowels
Q) 그 자음 중에 velum이 낮아지면 무슨 소리일까? : /ng/

*** Phoneme**: 각각의 개별적 소리 (철자와는 구분됨)
-> a combination of speech organ's actions: lips - p,b,m,f,v,w / tongue tip - θ,ð,t,d,s,z,ʃ,ʒ,l,r
/tongue body - k,g,ŋ,j,vowels / velum(lower) - m,n,ŋ / larynx(open:무성음) - p,f,θ,t,s,ʃ,k,h

2) Acoustics 음향 (Acoustics in **"praat"-음성 분석 프로그램**)

*** Praat의 기본사용법**
- new: 녹음 record -> save to list -> (창 끄고 본 창에서) view& edit
저장: save to list 후, sound untitled를 클릭하고 'save' -> save as WAV -> 이름 뒤에 ".wav" 붙여서 저장하기!
- open: 기존에 녹음된 wav 파일 불러오기
- sampling frequency에서 소리의 음질 결정. (ex. 44100 Hz : '1초'를 44100개로 쪼갠다는 의미)

① 소리 측정하는 요소 5가지

①-1. Duration(sec.) : select된 부분의 '기간'을 정확한 수치로 보여줌.
①-2. Intensity(dB) : 'show intensity'-> 노란 줄이 '소리의 강도'를 나타냄
①-3. Pitch(Hz) : 'show pitch'-> 파란 줄이 '소리의 높낮이'를 나타냄.
('pitch setting - pitch range': 남성일 경우 '65-200Hz', 여성일 경우 '145-275Hz')
①-4. Formant(Hz) : '모음을 구별하는 수치적인 지표' -> 빨간 점을 따라 생성되는 띠 (F1,F2,F3,F4..)
①-5. Spectrogram : 소리의 '스펙트럼'(빠르게 움직이면 고주파 ->위로 갈수록 /느리게 움직이면 저주파 ->아래로 갈수록)

② Vowel acoustics

: /a/소리를 녹음 한 후, 'show pitch'해서 나온 본인의 pitch의 수치 확인
-> 'new - sound - create sound as pure tone - tone frequency'에 본인의 pitch를 입력
-> "sine wave(=pure tone)"가 나옴 => 'play'하면 기존의 그래프와 '소리의 높이'가 같음.(wave의 주기가 같기 때문)
(*pitch의 수치: '1초'에 내 성대가 떨리는 횟수/ *기존 그래프에서 반복되는 가장 큰 wave: 성대가 한번 떠는 움직임과 일치)

Q) 시험

: ①constrictors(lips/tongue tip/tongue body) ②velum ③larynx // ④constriction location ⑤constriction degree

=> 이 요소들로 각 음 다 구별 가능해야 함.(미리 해 놓기)(이 음에서 다른 음을 바꿀 때, 어떤 요소가 변하는지)

****vowel acoustics**

- pitch
: 반복되는 가장 큰 wave가(repeating event=성대의 떨림)가 1초 동안 몇 번 반복되는지
= 1초에 내 성대(vocal cords)가 떨리는 횟수('Hz'의 개념) = frequency

- sine wave (=pure tone)
: 대표적인 반복되는 신호
: ①1초 동안에 싸인 웨이브가 몇 번 나오는지(frequency=진동수) ②wave의 '크기'(amplitude)
-> sine wave의 형태를 결정짓는 요소

- 성대의 떨림과 일치 = wave

- frequency 맞추면 소리의 높이는 똑같이 나오지만, quality는 다름(다른 소리 나옴)

****source**

: 첫 번째 소리에 비해 두 번째 소리는 무슨 모음이 발화되는지 알 수 없음.
두 번째 소리는 성대에 대고 바로 녹음 한 것.(마이크에 대지 않고, human voice source)
(어떤 소리가 만들어지는가는 입에서 어떻게 움직이느냐-입의 모양-에 따라 결정되는 것.
성대에서 내는 소리는 음의 높낮이는 다르지만 다 같은 소리)

****complex tone in spectrum**

: 사운드를 포함한 이 세상의 모든 신호(signal)는 다르게 생긴 여러 sine waves의 합으로 포현될 수 있다.
(sine wave는 가장 기본적 형태로, frequency와 magnitude/amplitude크기에 의해 결정됨)

- frequency : ③ > ② > ①
- amplitude : ① > ③ > ②
- sine wave ①
: frequency 적으나(slow), amplitude 큼 -> '저음'에 해당됨.
: 100Hz(1초에 반복되는 것이 100번 들어감)
- sine wave ②
: 200Hz(1초에 반복되는 것이 200번 들어감) -> sine wave①보다 2배 빠름.
- sine wave ③
: 300Hz(1초에 반복되는 것이 300번 들어감) -> sine wave①보다 3배 빠름.
- sum(①+②+③)
: complex tone, 복잡한 신호
: sine wave①과 반복되는 주기 똑같음(1초에 100번 반복 = 100Hz)

- 서로 다른 sine waves(simplex tones)의 합은 복잡한 신호로 표현됨
= 복잡한 신호, 소리는 단순한 sine waves의 합으로 표현될 수이다.

- (시험 나옴) (*수업자료의 표 확인할 것!)
: (x)time / (y)vlaue의 sine wave를 => (x)frequency/ (y)amplitude의 그래프(spectrum)로 변환 가능
  (어떤 시점에서 어떤 주파수 성분이 많은지를 분석하는 것이 'spectrum')

- spectrum
: spectrum ①+②+③ = spectrum ④
: ①+②+③ => "④" _ spectrum synthesis (합성)
  ④ => "① / ② /③" _ spectrum analysis (분석)

**practice with pure tone & spectrum
- pure tone = simplex tone = sine wave

- 모음 /아/소리를 녹음 후 똑같이 해보면(spectral slice, spectral analysis), 복잡한 그래프 나옴(complex sound)
  -> 확대해보면 패턴이 보이는 듯함. 그래프의 산(꼭점) 부분 간의 간격이 똑같아 보임.
  -> 등간격으로(배수로) 이루어진 그래프. 각각의 sine wave으로 이루어진 것.

- complex sound의 frequency는 합해진 sine waves 중 주파수가 가장 낮은 것과 일치
  (frequency가 가장 낮은 sine wave와 반복되는 주기가 같다)

- 첫 번째 꼭지의 frequency 대략 130Hz(frequency가 가장 낮음) = "pitch"와 일치
  (-> 다음 꼭점은 260Hz ->390Hz ... => 배수로 반복)

- complex tone은 여러 다른 simplex tones의 합으로 이루어짐.
  가장 느린 simplex tone의 frequency가 pitch와 일치함.
  우리 성대에서 몇 번 떨리는지 와도 일치함.

*"pitch"(음의 높낮이) = 가장 작은 sine wave(pure tone)의 " frequency(진동수)(Hz)"
  (반복되는 가장 큰 wave가 1초에 몇 번 나오는지)
= "성대(vocal cords)에서 1초에 몇 번 떨리는지"와도 일치함

*모음을 어떻게 만들까?
: 해당 모음의 pitch를 안다면, 그 pitch(가장 느린 sine wave의 frequency)에 해당하는
  sine wave를 배수로 반복해서 만들어 합하면 그 모음의 소리를 만들어낼 수 있음.

**Human voice source
: 모음 /아/ /이/가 다른 것은 우리가 '입모양'을 달리했기 때문. 성대에서 나는 소리는 같을 것.
  (pitch는 같다는 전제 하.)

**filtered by vocal tract
: 성대에서 나는 소리 바로 녹음 => 무슨 소리인지 들리지 않음

- source: 성대(larynx)에서 나는 소리. (항상 same)
  (tube 직전에서 나는 소리)
  : EGG(voice cords에서 직접 녹음한 소리)
- filter: tube가 어떻게 달라지는지. (*tube/vocal tract: 입에서 성대로 연결되는 관)
  : peaks/ mountains = frequencies VT likes = formants
  (첫번째 산맥 -> "F1" /두 번째 산맥 -> F2... // 'F1'의 'frequency' -> '첫번째 formant')
  valleys = frequencies VT does NOT like

=> source에서 filter를 어떻게 바꾸느냐에 따라 다른 소리가 남.
(입모양에 따라 어디에 산맥이 나타나는지 다름)
(콜라병에 바람을 불어넣어 소리를 낼 때 -> 콜라병의 모양에 따라 이미 어떤 소리가 날지 정해져 있음.
특정한 음의 높이를 특정한 입모양이 좋아할 수 밖에 없음. => 'formants')

- “source”를 spectral analysis하면 'spectrum' 그래프
  (모든 사람의 source는 다 같은 패턴으로 나타남)
: spectrum은 sine waves의 합.
: [F0 = the lowest pure tone의 frequency = fundamental frequency
      = pitch = the number of vocal cords vibration in a second = 제일 첫 번째 harmonic]
  (F0: frequency가 가장 작은 sine wave의 frequency/amplitude는 가장 큼)
  (*F1: 가장 첫 번째 산맥 peak)

: "frequency"가 harmonics이룸 -> 간격이 배음(배수로 무한대로 반복됨/x2, x3 ...)
  여성의 경우, **frequency의 첫 시작이 크고(pitch가 높음), 대신 간격이 큼(첫 frequency의 배수로 반복되므로)**
  남성의 경우, **freqeuncy의 첫 시작이 작고(pitch가 낮음), 대신 간격이 작고 배움의 수가 더 많을 것.**
            (여성에 비해 더 자주 반복 -일정 부분에서)
: "amplitude"의 경우, ‘gradually decreased’!!
: sine wave의 F0가 정해지고, 그 값의 배음의 합으로 source가 정해지는 것.

- "spectrogram"은 spectrum을 시간 축으로 늘여놓은 것. (입체)
: (x)time / (y)frequency

**synthesizing source
: 1000Hz까지 만들기 -> amplitude를 0.05씩 줄여가며, 100Hz씩 키우기

- combine to "stereo"
  -> 여러 sine wave를 더한 상태 아님.
     (한 파일에 있긴 하나)독립적으로, stereo로 존재하는 상태인 것.
     : **여러 개가 동시에 존재 /stereo <-> mono**
  -> 이걸 하나로 합하면, 하나의 complex wave를 만들 수 있음.

- convert to "mono" (합하기)
  -> complex tone
  -> 반복되는 패턴이 보임/ 반복 주기 :10개의 sine wave 중의 F0의 sine wave와 일치
  -> 이 소리는 100Hz와 높이가 똑같다고 인지.

  -> 부드럽게 가던 F0의 sine wave가 점점 한 쪽으로 뾰족한 모양으로 나타남.
  => ‘무한대’로 갈 경우, [peak 하나 있고, 0,0,0,0 , peak 하나 있고, 0,0,0....]의 패턴
     : “perse train”

- mono wave에서 일정부분 선택한 후, plot a spectrum(view spectral slice)
  (선택한 그 일정 시간에서의 spectrum)
  : 10개의 sine wave가 보임 -> frequency가 100Hz에서부터 시작함 /amplitude가 gradually decreased.

**formants
: F1, F2만 있으면 모든 모음의 특징이 구별됨. (F3, F4는 무시) (F1, F2는 모음마다의 서로 다른 도장같은 것.)

**vowel space
: 서로 다른 모음이 서로 다른 입모양 지님.
: f1,f2로 위치시켰을 때, 입의 위치와 똑같음.
: 사진처럼 입이 왼쪽을 보고 있다고 생각했을 때,
  = > F1: 모음의 '높낮이(hight)'를 결정 -> high / low
      F2: 모음의 '위치'를 결정 -> front / back

----------**코딩 본격적으로 시작**----------
- 코딩: ‘자동화‘
  -> 왜 자동화해? 똑같은 일이 계속 반복되므로, 자동화 하는 것이 이익이 됨.
  -> 코딩도 마찬가지. 직접 다 할 수 있으나 반복되는 일 자동화 하면 훨씬 편리
     : 핸드폰, 컴퓨터에 있는 모든 것들이 코딩으로 이루어져 있음.
- 컴퓨터 언어도 여러 가지 존재.

- 모든 언어는 단어와 문법으로 이루어짐. (존재하는 단어를 결합하면 끝)
- ‘단어’란 뭘까? 단어의 특징, ‘정보’(의미)를 담는 그릇.
  (하나의 그릇 존재, 그곳에 하나의 정보만을 담을 수 있음. 사과를 담으면 사과가 되는 것.)

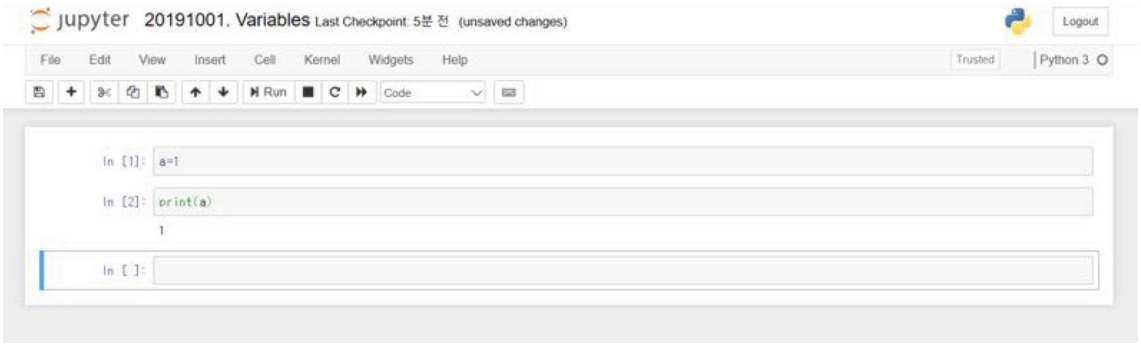
- 어떤 단어를 선택해서 어떻게 combine할 것인지 -> 문장 만들어 의미 표현 가능.
  -> 컴퓨터 언어에서 단어에 해당하는 것이 '변수(variable)'
     변수에 정보를 담고, 기계가 사람과 소통하기 위해선 문법이 필요. (기계의 문법)

**컴퓨터 언어의 "문법"
1) 변수라는 그릇에 정보를 넣는 것.
   => variable assignment
2) 'conditioning'에 대한 문법이 필요 : 'if 문법'사용
   => if conditioning (조건문 if)
3) (자동화에서 가장 중요한) 여러 번 '반복'하는 것: 'for 문법'
   => for loop (반복문 for)
4) 함수(가장 중요) -> def
   (3번까진 개별적 문법)
   많은 내부적인 명령들(1,2,3번)을 한꺼번에 '입력과 출력'으로 packaging하는 것.
   (입력이 들어가서 바뀌어 출력되도록 하는 것)
   (ex. 자동차에서 '입력'을 주어 빨리 달림-> 출력
        두 개의 숫자를 넣으면 첫 숫자와 두 번째 숫자 사이의 숫자들의 합을 구하는 함수

** '재사용, 반복사용'이 가능함(코딩의 필요성)

*** variable
- 컴퓨터에서 정보를 담는 역할 (숫자, 글자) , 정보를 주는 '단어'의 역할

1) 숫자
* a = 1 (입력하고 'run' 버튼 클릭)
  => '=' :오른쪽의 정보를 왼쪽의 variable로 assign한다는 의미 (오른쪽이 늘 정보, 1라는 정보를 a이라는 variable에 집어넣는다)



* 'print( )'함수에 변수 a를 입력하면('run버튼' 클릭) a의 값을 출력할 수 있다.
  (print: 입력한 변수를 출력해주는 함수)
  (-> '1'은 print의 결과일 뿐 , 그 줄은 cell이 아님.)

* 아나콘다: 유용한 함수들을 모아둔 것. (파이썬 + 좋은 함수들)

* cell 생성 & 삭제 : 클릭 하고 b 누르면 아래에 새로운 cell 만들어짐. a 누르면 위에 새로운 cell 만들어짐. x 누르면 cell 삭제

2) 문자
문자는 반드시 ' '를 사용해야함 (single quote/double quote 둘 다 상관없음)
  => ''를 사용 안하면 변수, ''를 사용하면 정보(문자)로서의 text

* '실행' 하는 건 cell 클릭하고 'shift+enter' 누르면 됨.

-> 함수 print( )로 확인 -> love가 출력됨.

```
In [12]: b= 'love'

In [13]: print(b)
         love

In [14]: love = 2

In [16]: b = love

In [17]: print(b)
         2
```

=> love = 2
: love라는 변수에 숫자 2를 넣는 것.

=> b= 'love'
: 문자 love를 b에 assign한 것.

=> b = love
: b에 love가 가지고 있는 변수의 내용(2)을 넣은 것.
-> print(b) 하면 2가 나옴.
-> 만약, 해당 변수를 assign해 놓지 않았으면 error!

```
In [14]: love = 2

In [18]: b = hatred
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-18-f18fae117903> in <module>
----> 1 b = hatred

NameError: name 'hatred' is not defined
```

(-> hatred란 이름은 한번도 정의되지 않았음!!)

-> hatred를 quote(따옴표)하면 변수 b에 hatred라는 문자가 부여되는 것.
print(b)했을 때 hatred가 출력됨.

```
In [19]: b = 'hatred'

In [20]: print(b)
         hatred
```

```
In [21]: a = 1
         b = 2
         b
         c = 3
         c
Out[21]: 3

In [22]: c
Out[22]: 3
```

-> print(c)라고 하지 않아도, 가장 마지막에 변수를 하나(c)만 치면 print한 값이 출력됨.

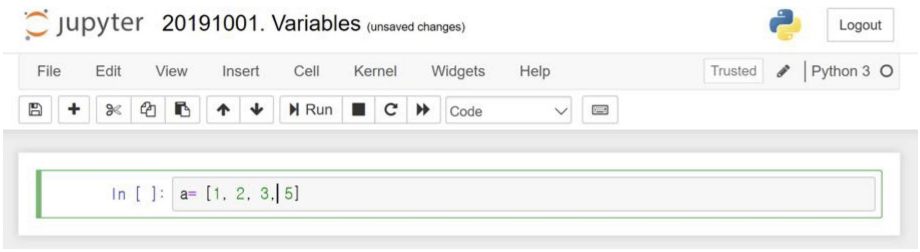
```
In [23]: a = 1; b = 2; c = 3

In [24]: print(a); print(b); print(c)
         1
         2
         3
```

-> ; (semicolon)을 사용하면 여러 줄을 한 줄로 사용 가능. (print도 마찬가지)

=> 지금까진 한 변수에 한 '숫자나 문자'를 넣음

** List
: 한 변수에 여러 정보를 넣고 싶을 때 (한번에 모아서 넣는 것)
: 숫자, 문자 다 가능.
: 대괄호[] 사용
(대괄호 대신 소괄호()를 써도 됨 : Tuple -> List와 기능 똑같음)



-> a에는 list로서 1,2,3,5가 들어가 있는 것.

** Type
: a에 들어간 것이 무엇인지 확인하기 위해선 함수 'type(변수)' 입력.

```
In [25]: a = [1, 2, 3, 5]

In [26]: type(a)
Out[26]: list
```

**숫자형
1) 정수형 : int

```
In [27]: a = 1

In [28]: type(a)
Out[28]: int
```

-> a=1을 type(a)해보면 'int' 나옴 (number라고 나오지 않은 것을 보니, 더 세분화되어 있을 것)
=> 정수형(integer): 정수(양의 정수, 0, 음의 정수)

2) 실수형 : float

```
In [29]: a = 1.2

In [31]: type(a)
Out[31]: float
```

-> a=1.2를 type(a)해보면 'float' 나옴.
=> 실수형(Floating-point): 소수점이 포함된 숫자

**문자형 : string

```
In [32]: a = 'love'

In [33]: type(a)
Out[33]: str
```

-> a = 'love'를 type(a)해보면 'str' 나옴.
*만약 a='love'를 실행하지 않고 type(a)를 입력하면 float(실수) 뜸.(전의 입력값에 대한 출력값)

```
In [37]: a = [1, 2, 3, 5, 'love']

In [38]: type(a)
Out[38]: list
```

-> list에 반드시 숫자만 들어갈 필요 없음. 문자도 가능

```
In [39]: a = [1, 'love', [1, 'bye']]

In [41]: type(a)
Out[41]: list
```

-> list 안에 list가 들어 갈 수 있음.

-> a라는 list 속에 몇 개의 item이 존재?

: 3개/ 1이라는 int, love라는 str, [1, 'bye']라는 list -> 3번째 list 속의 item은 1이라는 int, bye라는 str.

**Tuple

: List의 대괄호[] 대신 소괄호() 사용하면 Tuple. (List와 기능 똑같음)

```
In [42]: a = (1, 'love', [1, 'bye'])

In [43]: type(a)
Out[43]: tuple
```

=> tuple이 list보다 좀 더 보안에 강함.(바꾸기 힘듦)

**dictionary (dict) 사전

: 중괄호{} 사용, comma(쉼표),로 몇 개의 item인지 구분.

: '단어(표제어) + 단어에 대한 설명'의 쌍으로 이루어짐 -> 따옴표quote('')와 colon(:)으로 표현.

```
In [44]: a = {'a': 'apple', 'b': 'banana'}

In [45]: type(a)
Out[45]: dict
```

-> a라는 표제어와 그에 대한 설명, b라는 표제어와 그에 대한 설명

-> dictionary에 2개를 넣어놓은 것(,콤마에 따라)

반드시 string만 들어갈 필요는 없음.