

## Part 2

영어뿐 아니라 모든 언어는 자음과 모음으로 되었다

‘ g a p ’ : 그 가 자음. (소리)

sh z - 진동의 유무 ( she vision)

j(y) - yearn yaught (요트) / ear과 다른소리 (모음으로 시작 )

f v - 목이 떨리는지 안 떨리는지 / f - voiceless. v- voiced

모든 모음은 voiced.

mn ng - voiced / nasal

모든 자음을 voice voiceless 로 나눠보기.

입술 두 개 쓰는 - p b

혀랑 윗니 - th

**phonetics** 음성학

-인지적인 것 ) phonology 같게 생각되는 부분 cognitive

-phonetics ) 더 물리적이고 늘 차이가 있는 . physical

- a study on sound system

- articulatory ) 어떻게 소리가 나올까. 폐가 고기압을 만들어 push 하고 공기를 보내줌. -

성대에서 소리가 만들어짐 - 바람은 파도와 비슷. / whisper 할때는 기문을 완전히 염.

성대에서 기문을 막고 있어서 강한 바람에 의해 펄럭펄럭 움직이는걸 진동으로 느낌.

>어떤 메커니즘에 의해서 바뀌는지 . / ‘아’에서 ‘이’로 바뀌는건 입모양. (혀의 위치. 턱의 높  
낮이는 고정됐어도 입속 모양을 바꿔서 소리 바꿀수)

- 영어는 stress 강세가 있다 .

acoustic : 공기가 어떻게 타고 가는지. 물리적.

auditory : 공기가 물결치듯이 타고 오는 소리를 귓바퀴(소리 증폭)를 통해서 듣는 것. / 어떻  
게 듣는가. 귓바퀴 타고 들어가 고막 ear drum 이 움직이고 치는 것이 청각세포로 전달.

## Articulation

인강 후두( larynx)

alveolar - t d s z m l

epiglottis 꿀떡하는 순간 기도로 가는 길을 막는다.

vocal tract

nasal tract

m 소리르 낼 때 oral tract 는 막혀 있고 nasal tract 은 열려있다.

nasal tract는 velum 이 올라가면 막힌다. - 모든 모음/ 비음을 뺀 모든 자음

코로 숨을 쉴 때 velum 은 lowered, nasal tract 는 열림.

oro - nasal process

phonation process - voiced(막혀서 성대가 진동)/ voiceless ( 공기가 그대로 통과)

## <9/19> part 3

phonetics- 조음 음향acoustics 청각auditory-어떻게 받아들이는가

articulators 크게 세 개.

아 할 때 그 동영상같은 현상이 일어남  
larynx가 스피치에서 사용될 때 어떻게 소리가 분류되는지  
-voiced ( can feel vibration ) / voiceless

oral-nasal process in velum  
m n ng : velum lowered  
숨을 쉴 때 velum이 내려가- 코로가는 길이 생김

Articulatory process in lips- b / tongue tip(혀 앞쪽) - t l s/ tongue body  
실질적으로 어떻게 움직이는지 xray 찍은 사진. : 아파-tongue은 메이저로 움직이지 않고 입  
술 두 개가 움직임/ 아타 - 혀끝을 침/ 아카 - 혀의 뒷부분이 치고 내려옴

#### Control of Constrictors

-lips tongue tip/body를 constrictors 이라고 부름.  
얼마큼 constriction이 일어나냐 따라 CD를 컨트롤 할 수 있고 / 위치적으로 조금 앞에서할  
건지 뒤에서 할건지 location 컨트롤가능 / tongue tip을 가지고 조금 막을건지 많이 막을건  
지 앞 뒤로 갈건지에 따라 달라짐  
-constrictor는 CL CD에 따라 specify  
CL - 앞뒤 lips가 조금 앞으로 가면 b p (bilabial), 조금 뒤로 가면 f v(labiodental)/  
tongue body 가 조금 앞으로 가면 yarn 할 때 y 조금 뒤로 가면 g / tongue tip - th 윗  
니를 침 , 조금뒤로가면 alveolar 필수도 sh 조금 더 뒤. r 은 더 뒤 (4가지)  
CD - 상하 얼마큼 upper part를 hit하나에 따라 CD가 결정됨.  
: d - upper part를 완전히 쳐서 stop- p t k / 살짝 틈을 주는거 - s fricative / 완전히  
띄우는 approximants - 4개. r l w j / vowels 모음은 막힘이 없음.  
CD의 관점에서 자음은 3가지 종류 - stop/ F/ A  
Q. ng 은 어디에 해당? - stop 완전히 막혀서 ( m n ng 다 stop )  
fricative - s z f v th sh 등..

+ velum을 내리냐 올리냐 / larynx을 올리느냐 닫느냐  
Q. velum raised, glottis opened , C - tongue tip . CL - alveolar, CD- stop  
▶ t  
모든 모음은 반드시 constrictor로서 tongue body만 쓴다.  
Q. 모음과 같은 constrictor을 쓰는 자음의 예?  
>k / velum lowered - ng 성대가 올림. glottis closed

#### phonemes

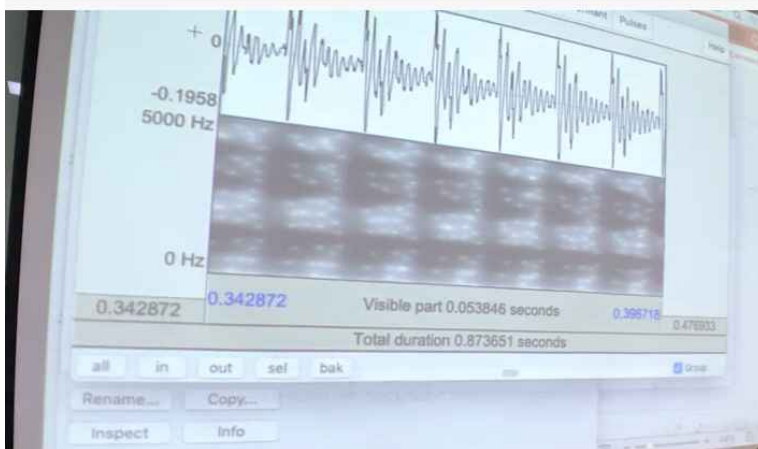
- lips 가 active하게 움직이면 > p b...  
- 모든 모음 자음을 specify 할 수 있어야.

#### Pratt

- object ( 녹음하기/ 녹음된거 불러오기)
- 파란부분이 자음.
- frequency 의 관점에서 spectrum analysis 할 수 있음.
- duration/ pitch/ intensity measure할 수 있음.
- 회색바탕이 spectrogram.
- 까만띠가 4개 보임. 띠를 formant라고 함. 빨간점 / 쉼 밑에 있는 띠가 첫 번째 formant f1 / f1 f2 가 뭐냐에 따라 모음이 뭔지 결정. 어떤 모음이든지 간에 모음을 구별하는 수치적 지표로서 formant가 쓰인다. formant값이 뭔가에 따라서 이 모음은 뭐다라고 얘기할 수 있음.
- pitch 여성/남성의 목소리가 에 따라서 pitch setting에서 range를 남자 목소리라 치면 65~200. 여자는 더 위로해서 해야 잘 측정이 됨.
- formant 커서를 대면 빨간부분이 수치적으로 얼마지도 볼 수 있음.

### vowel acoustics

new - record - view



밑에 파란선이 pitch

확대를 해보면 젤 큰 wave가 반복. 이 반복되는게 larynx 가 빨리 움직이는 가와 같음. 1초에 몇 번 떠는가. 큰 wave 하나가 한번 떠는.

-pitch 가 129 Hz : 1초에 내 성대가 129번 떨린다.

>반복되는 wave하나 선택해서 확대해서 drag해보면 duration이 0.007852: 1초에 이게 127번 들어감. / 1나누기 저 숫자. / 나의 pitch는 내가 1초에 몇 번 움직였냐에 대한 계산.

-new - sound - creat sound as pure tone을 해보면, tone frequncdy를 127 로 입력 (자신의 피치와 같은걸 ) . > view & edit > 사인 곡선 > 소리 들어보면 ( modify - multiply하면 크게 들림 100정도 ) 높이가 똑같이 들림.

>>127로 파도가 똑같기 때문에 파도의 주기가 똑같기 때문에.

#### Part 4

constrictors - lips Tongue top Tongue body

velum

larynx

p 라는 소리 specify - lips. ( CL - BL , CD - Stop ), velum raised, larynx open.

/b/ - larynx closed 로 바뀔 위에서.

/d/- tongue tip. ( CL - alveolar , CD - stop )

/z/- tongue tip ( CL- CD-fricative )

/n/ - tongue tip ( CL- alv CD- stop ) . larynx closed. velum lowered. (nasal tract)

/? open

>>시험!

**Praat** 시험에 나왕

**vowel acoustics** (젤 중요!!)

repeating event가 어느정도 시간이 되는가 1초동안 몇 번나오는가가 pitch를 제는 방법.

( measure의 단위는 Hz)

sine wave 가 대표적인 반복되는 신호. -

1초라는 구간이 주어지면 1초동안 sine wave가 얼마나 나오느냐에 따라 주파수를 알 수 있음

sine wave)

주파수 frequency - 1초당 몇 번 반복되는가.

sine wave의 크기에 의해서도 결정됨.

> 이 두가지에 의해 형태가 결정됨.

크게 반복되는게 성대의 떨림과 일치.

a를 녹음하고 확대했을 때 성대의 떨림과 일치.

sound quality 는 다르지만. 주파수, 크기?는 같지만.

## Source

장치로 성대에서 바로 직접 녹음을 하면  
소리가 어떻게 만들어지는가는 입에서 결정.  
아, 이 같은 거는 입모양에 의해서 바뀔.

## Comple tone in spectrum

Signal .

sine wave 가 리드믹하게 반복되는 가장 기본적인 시그널의 형태.

를 결정짓는게 pitch frequency, 소리의 크기magnitude에 따라 형태가 결정됨.  
이 세상에 존재하는 모든 사운드를 포함한 시그널은 여러 다르게 생긴 sine wave의 결합으로  
표현된다. (모든 신호는 조금씩 다른 sine wave 의 합으로 표현될 수 있다 . ) 주식시장의 흐름도  
신호라고 생각할 때 모든 신호는 싸인 웨이브들의 합.

첫 번째 sine wave, 크기도 크고 frequency 는 작음. (천천함 ) 저음에 해당.

1초에 반복되는게 100번 들어감 . magnitude 쥔 큼

두 번째, 200hz 1초에 200 번 들어감 . 첫 번째 보다 2배 빨라 . magnitude 쥔 작음.

세 번째, frequency 상대적으로 높음. 첫 번째 보다 3배 빠름.

세 개를 합하면(더하기)/복잡한 신호로 만들 수 있다. 복잡한 신호는 단순한 다양한 사인웨이브들의 합에 의해 만들어질 수 있다.

결과적으로 나온 복잡한 신호를 볼 때, 반복되는 주기는 첫 번째 사인과 같음. 1초에 백번 반복. - complex tone

하나하나를 simplex- simplex tone(sine wave) 마지막엔 complex tone- sine wave x

하나의 simplex tone을 더 단순하게 표현한건 오른쪽거.( x축은 frequency y축은 amplitude >>스펙트럼 spectrum ) /amplitude=magnitude

왼쪽의 x축은 뭘까 ? 시간 /y축은 그냥 value. 숫자값 시험 !

-오른쪽으로 표현하면 x축은 frequency y축은 amplitude로 변환.

오른쪽의 그래프 ( spectrum이라고 함 ) 으로 변환할수 있어야.

spectrum 은 이퀄라이즈 와 같음.

spectrum 은 어떻게 이루어져 있을까?

우리 주변의 소리는 복잡한 형태. complex tone

위에서 밑으로가는. 합쳐서 밑에걸로 만드는데. 합성 synthesis. 합쳐서 마지막처럼 만드는데 밑에를 쪼개는 걸 analysis. ( spectrum 으로 쪼개는 )어떤 것들의 합으로 이루어졌는가 보는거.

## Practice with pure tone&spectrum

sound- creat pure tone - tone frequency 440(디폴트 값) amplitude 1 - view edit

확대해보면 sine wave

value 들의 최저값 -1 최대값 1 (amplitude를 1로 했기 때문에 / 0 ~1이 A 값)

주기가 1/440 초

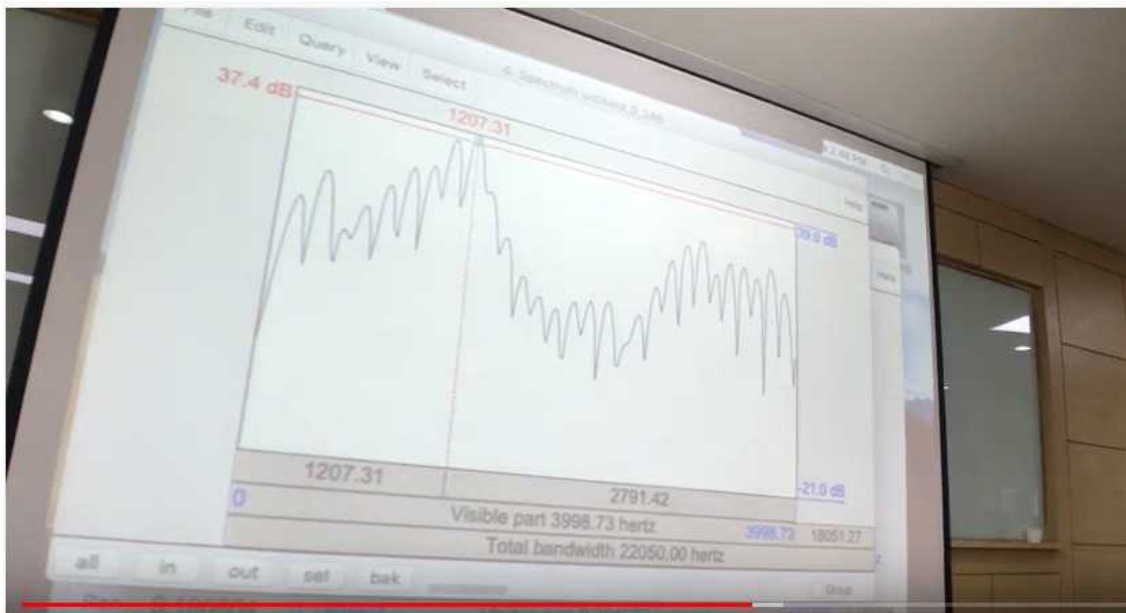
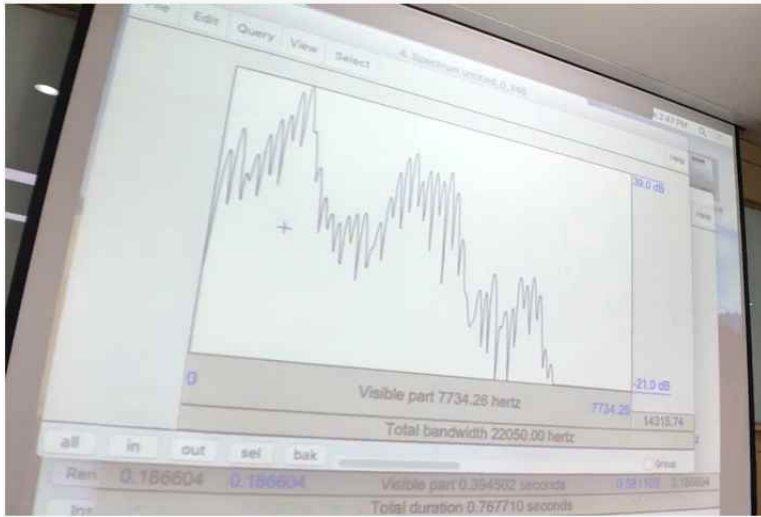
spectrogram : 스펙트럼을 time으로 visualize한거 /시간축으로 계속 늘어놓은거

조금 select해서 spectrum - view spectral slice - 확대

440에 피크가 나옴. x축 frequency축에서 440이 되는 포인트.

440hz를 가진 sinewave를 만들어서 spectral analysis를 해봤더니 440을 가진게 있더라.

아를 녹음해서 spectral analysis 해보면 ( x축 frequency y축 amplitude)



아까는 440 하나만 있었는데 많음. / 저주파에 성분들이 많음

-확대해보면 / 점 사이의 간격이 같음.

1206 나누기 9 해보면 133. /

이콜라이즈

- 등간격으로 되어있음. ( 130, 260, 390...)

133HZ 와 일치.

첫 번째 sine wave의 130은 나의 pitch인 133hz와 일치.

> simplez tone의 합을 여러개 해봤을 때 반복되는 패턴이 여러개. 젤 낮은 주파수의 그것과 일치. /아의 pitch는 젤 작은 simplez톤의 frequency와 일치하는.

\*마지막 패턴은 첫 번째 반복되는 패턴과 같음. 젤 작은 simplex frequency 133 와 우리말

의 pitch와 일치. vocal folds가 1초에 몇 번 떨리느냐와도 일치.

+아를 녹음하고 analysis를 해봤더니 단순한 simplex sine 들의 합임.

pitch - 1초에 (반복되는 주기가) 몇 번 반복되는가. vocal fold가 떨리는 주기와 같음(1초에 몇 번 떨리느냐)

모음을 어떻게 만드느냐. - pitch에 해당되는 frequency를 안다면 S을 만들고 배수로 넘어가면 됨. (100,200, 300...hz인 sine wave를 다 합하면 아라는 소리가 나오는 )

진동수 - frequency . 반복되는게 1초에 몇 번 자리잡고 있는가. 단위는 Hz.

음의 높낮이(pitch) 는 켈 작은 sine wave의 진동수와 일치한다 !

### human voice source

모음중에서도 아 이 가 다른 것은 입모양으로 달라지는 것.

성대에서만 녹음을 하면 똑같은 소리.

성대에서 나는 소리를 그대로 캡처했을때의 소리를 source. - larynx에서 나는 소리

filter - 튜브 가 어떻게 달라지느냐 . / source에서 filter를 어떻게 바꾸느냐 에 따라 아 이 소리를 만드는 것.

source를 가지고 spectral analysis를 할 수 있음. > 오른쪽 그림.

아를 녹음했을때보다 훨씬 깨끗. smoothly decreasing 모든사람들의 source의 패턴.

첫 번째 주파수F0 와 피치 일치.

sine wave들의 합

Fo - fundamental frequency. amplitude가 크고 점점 작아짐.

곱하기 2, 3되는게 harmonics라고 부름.

115, 230.. 거리가 여자라고 치면 첫시작이 더 높아 더 음성음성.

/만 hz까지 남자가 갖는 배음의 숫자가 여자보다 많음. (이런식으로 시험)

### Filtered by vocal tract

배음의 구조가 안깨졌음. (114, 곱하기 2 3 은) 그대로 유지.

amplitude의 패턴이 깨짐. 지멋대로 왔다갔다 A가 smoothly decrease 하는게 아니라

human voice source 슬라이드

spectrogram 읽는 방법 :왼쪽그림) wave와 쌍을 이룸. x축이 time.( sine wave도 ) y축은 frequency / 까만게 크기가 센거. 밑에는 까맣고 위로 갈수록 열어짐. low F 저주파 로 갈수록 에너지가 크고 high 고주파 로 갈수록 에너지가 약함 ( 오른쪽 그림에 low F에서 에너지가 크고 high로 갈수록 작아짐 )

아 로 녹음한거 뒤죽박죽인게 왼쪽그림에서 나타남. ( filtered by vocal tract )

0926

### Part 5

모든 소리는 complex sine (pure tone)

spectrogram 의 x축은 시간 y축은 frequency ( 시험 !!)

특정구간을 선택해서 spectral analysis / 어떤 성분이 많은 지 보는게 spectrum

vocal tract ( 입모양 ) filter 역할을 해 다양한 모음소리를 낼 수 있음.

등간격으로 harmonics가 유지

egg- voice source에서 직접 녹음한거

까만게 블록티어나온. ( mountain 과 valley )

모든 사람들의 '아' 의 패턴은 똑같이 나옴.

첫 번째 산맥에 해당되는 주파수 : 첫 번째 formant / 두 번째는 두 번째 formant....

어떤 특정한 음의 높이를 특정한 입모양은 좋아하게 됨. - formant와 일치.

어떤 산맥은 '아'라는 입모양이 좋아하는 산맥들이 있고, 어떤 소리의 높이들은 싫어하는 - valley

harmonics 가 되어 나는 소리 - 기타 소리 .

'라'를 치고 analysis를 해보면 배음으로 됨. - 사람의 성대에서 나는 소리 source 랑 똑같은 음.

기타소리는 - complex tone /

pure tone이랑 높이는 같음.

wave form 은 x축이 시간. y축은 value.

Practice with pure tone & spectrum

y축은 frequency 하는거 시험에 나옴

입 모양. 필터 역할을 함으로써 다른 모음을 소리 낼 수 있다.

Synthesizing Source

- voice source 직접 만들기 - 여러개 만들어서 합하기 - 10개

- 100hz, 200hz ... 1000hz / amplitude를 0.05만큼 낮춰.

- spectrogram 이 가장 낮은 부분에서 에너지가 까맣.

100, 200, 300hz 다 똑같은 음임.

- 다 선택해서 combine to stereo - 하나의 object로 만드는.

- view&edit 하면 10개가 다보임. sine wave 가 달라지는게 보임.

- 들어보면 pure tone 보다는 사람 목소리와 비슷.

>여러가지 sine wave를 + 안한상태. 독립적으로 stereo로 존재. 다 합하면 complex wave.

여러 개가 동시에 존재 - stereo / 그 반대는 mono ( 더하기로 합하는 )

-convert to mono - view&edit - wave 는 complex tone

-반복되는 패턴은 F0와 일치. / 소리가 100hz와 일치. 높이가 같다고 인식.

-이걸 무한대로 1100 1200 계속 합하면 peak 하나 0000 peak 하나 000.... - pulse(하나 뻗어 나온거 ) train?

- mono wave 조금 선택 (반복되는거 3개정도) - plot spectrum - spectrum slice - 그 순간에서의 spectrum ( 10 개 gradually decreasing )

\*tube. vocal tract 가 들어가면 specturm 은 산맥이 존재 ( 블록 튀어나오고 들어가는 )

도장 찍는 역할을 filtering / 첫 번째 블록 튀어나온게 first formant 두 번째게 second fromant / f0 은 켈 첨에 나오는거의 frequency / formant 는 피크의 frequency



f12 만 있으면 모든 모음 구분 가능. / 서로 다른 모음은 서로 다른 입모양. / 각각의 f12를 그래프로 나타낼 수 있음. - f2 가 x 축 1를 y축.

위에서 밑으로 Front back / f 12 가 입의 위치와 같음.

f1은 모음의 높낮이 결정. 혀의 높낮이. f2는 front back을 결정 backness

new - sound - create sound from vowel editor

영어 아 와 한국어 아 - 영어가 더 back 하고 더 low.

클릭하는 만큼 소리의 길이가 남. 클릭을 하고 drag를 하면 이중모음이 나옴.

“영어뿐 아니라 모음이 어떻게 만들어지는가의 원리. ”

<10/1>

코딩 : 자동화 - 똑같은게 반복될 때

컴퓨터 안에 쓰는 모든 것은 코딩으로 이루어짐

컴퓨터 language 와 사람의 언어는 같음.

language들의 공통점 : 문법, 단어.

모든 언어는 단어가 있고, 이를 어떻게 combine 하나.

단어 - 의미. 정보가 포함. 정보를 담는 그릇. ( 그릇 하나에 정보로서의 사과를 담으면 이 단어는 사과가 됨 )

컴퓨터 단어에 해당되는 것이 변수./ 정보를 담는 그릇이 변수로서 필요하고 기계한테 communication을 하는 문법이 필요.

컴퓨터 문법

1) 변수라고 하는 그릇에 정보를 넣는 것. variable assignment

2) 자동화 기계화라고 생각할 때 , 조건이 필요. conditioning에 대한 문법. if conditioning

3) 여러 번 반복하는 것. for? 몇 번 동안 반복하라

4) 쯤 중요 ! 함수를 배워야. - 어떤 입력을 넣으면 출력이 나오는 것.

많은 명령들( 1), 2), 3). 을 한꺼번에 입력과 출력으로 패키징 하는 것

수치적 함수의 예 - 두 개의 숫자를 넣으면 그 시작과 끝까지 합하라 하는 함수.

자동차 - 핸들을 틀면 자동차가 나아가는

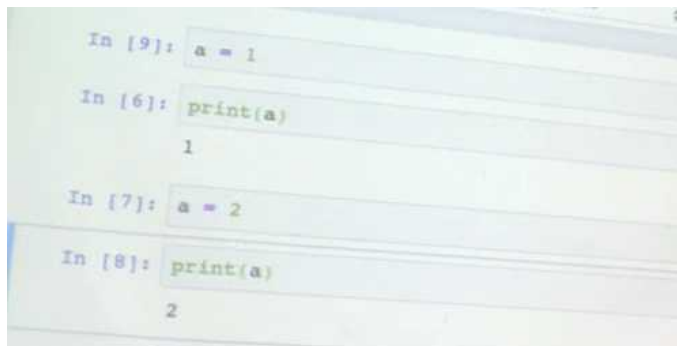
함수 속에 들어가는게 1)2)3) 이 될 수 있고 함수 속에 함수가 들어갈 수도.

anaconda prompt - jupyter notebook

컴퓨터 디렉토리 - 어디에 위치해 있다. / new - python3

변수가 정보를 담는 역할.

정보의 종류 - 숫자, 문자



=은 같다는게 아니라 오른쪽의 정보를 왼쪽에 있는 variable 로 assign 한다. - 순서가 바뀌

면 안됨. / 1이라는 정보를 a 라는 변수에 넣는다. / run을 하고 1이 변수에 들어갔는지 확인 - print(a)

print는 누가 만들어 놓은 함수 - 어떤 변수를 그 안에 넣으면 스크린에다가 그걸 print뿌려주는. / python의 모든 함수는 누가만들어놔야하고 , 내가 만들 수도 있고

함수이름을 치고 ( ) 안에다가 입력을 해주면됨.

print라는 함수의 입력은 a. - 입력이라고 표시해주는건 괄호. > 1을 print 해줌.

1이 떠있는거는 저 라인은 cell이 아님.

a=1에서 다음셀이 또 a=2 를 치면 1은 없어짐 .

파란색으로 바뀔 때 b를 치면 below에 cell 하나 더 만듦.

a를 치면 above 에 cell을 하나 더 만듦. 지우고 싶으면 x

문자를 넣는 방법)

b= love 하면 안됨.

문자는 반드시 quote 해줘야.

실행의 단축키는 shift enter

b = 'love'

print(b)

love

love라는 변수에 숫자 2를 넣는

love = 2

```
In [7]: a = 1
In [4]: print(a)
1
In [5]: a = 2
In [8]: print(a)
1
In [11]: b = 'love'
In [12]: print(b)
love
In [15]: love = 2
In [16]: b = love
In [17]: print(b)
```

b= love

b에다가 love가 가진 변수의 내용( 2) 을 넣는 것.

print c 라고 안하고 c만 쳐도 보여줌.

a=1; b= 2; c= 3

```
In [19]: a = 1  
         b = 2  
         c = 3  
         c
```

```
Out[19]: 3
```

```
In [20]: c
```

```
Out[20]: 3
```

print (a) ; print(b) ; print (c

1

2

3

세미 콜론을 하면 한줄에 다 넣을 수 있음

```
In [23]: a=[1,2,3,5]
```

```
In [24]: type(a)
```

```
Out[24]: list
```

```
In [25]: a=1
```

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

```
Out[26]: int
```

```
In [27]: a=1.2
```

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

```
Out[28]: float
```

```
In [29]: a='love'
```

```
In [30]: type(a)
```

```
Out[30]: str
```

[] 하면 여러 숫자를 한번에 넣을 수 있음: list.

a에는 list 로서 1235가 들어가 있음.

a 에 들어간 data의 유형이 list 인지 그냥 숫anzi 알려면

> list형태의 타입인지 알려면 함수 type

정보로 들어갈 수 있는게 숫자-int/float 문자

문자를 넣을때는 string

list 에 반드시 숫자만 들어갈 필요는 없음. int, str, float 이든 집합으로 한번에 넣은게 list.

list속에 list도 들어갈수있을까/

- int, str, list( int, str) 가 들어가있는.

[]대신에 () 넣으면 tuple= list 와 같음.

tuple이 더 보완에 강함.

dictionary에 쉼표에 따라 두 개가 들어있는. { } 를 씀.

{ }를 써야 dict이고 쉼표로 하고. : 콜론으로 되있음.

표제어와 설명의 쌍, 콜론으로 엮어줌.

```
In [31]: a=[1,2,3,5, 'love']
```

```
In [32]: type(a)
```

```
Out[32]: list
```

```
In [37]: a=(1, 'love',[1,'bye'])
```

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

```
Out[38]: tuple
```

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

```
In [40]: type(a)
```

```
Out[40]: list
```

```
In [41]: a={'a': 'apple','b':'banana'}
```

```
In [42]: type(a)
```

```
Out[42]: dict
```

str 말고도 들어갈 수 있음.

<10/8>

암기 x

variables > string> syntax> numpy

variable - 숫자 , 문자

어떤 변수속에 정보가 들어있을 때,

a = 1

b = 1

c= a + b 하고 실행하면

c

> 2 나옴

-a 와 b 정보를 가져와 c에 담은 것.

a = [1,2]

b = [3, 4]

c = [ a{0} + b{0} ]

c

>4

여러개가 있을 때 취합해야될 때, a 의 0 번째, b 의 0 번째.

list 에 acces를 할 때 a 중에서 어떤 값 만 가져올 때 대괄호 해서 가져온다.

세미콜론은 두줄에 적을거를 한줄에

a 에 1을 넣었을 때 a의 type 은 'int'

float 이라는 함수는 어떤 변수가 들어오면 float 타입으로 바꾸기. a를 float로 바꾸기. 소수 점이 있는 숫자로 바뀜.

in [19]: float 인데 int로. print (a) 라고 한다면 1.

in [20] 대괄호에 들어가는 것은 index - 내부적 정보를 부분적으로 가져온다

```

In [17]: a = 1; print(type(a))
<class 'int'>

In [1]: a = 1; a = float(a); print(type(a))
<class 'float'>

In [19]: a = 1.2; a = int(a); print(type(a))
<class 'int'>

In [20]: a = '123'; print(type(a)); print(a[1])
<class 'str'>
2

In [21]: a = '123'; a = list(a); print(type(a)); print(a); print(a[2])
<class 'list'>
['1', '2', '3']
3

In [22]: a = [1, '2', [3, '4']]; print(type(a)); print(a[0]); print(a[1]); print(a[2])
<class 'list'>
1
2
[3, '4']

In [23]: a = (1, '2', [3, '4']); print(type(a)); print(a[0]); print(a[1]); print(a[2])
<class 'tuple'>
1
2
[3, '4']

```

2라는 문자를 가져온 것.

a = 123 ( ' '를 빼고 print(a{1}) 하면, error 이 나옴 )

>하나밖에 없으면 0번째 꺼지 1번째게 없음.

list 함수가 1,2,3 쪼개서 list로 만들어줌.

list 화 한후 2번째 걸 가져와라 - 3이 문자로서 나옴.

dict (표제어 : 내용 ) 의 정보를 access 할때는 abc를 index의 수단으로

“a” 에 해당되는걸 가져와라

in [22] : print(a{0}) 숫자로서 1을 print 한 것./ a 의 첫 번째것은 문자로서의 2. /

```

In [24]: a = {"a": "apple", "b": "orange", "c": 2014}
print(type(a))
print(a["a"])
<class 'dict'>
apple

```

```

In [27]: a=[(1,2,3), (3,8,0)]
print(type(a))
a
<class 'list'>

```

```

Out[27]: [(1, 2, 3), (3, 8, 0)]

```

in [ 24 ] : 정보가 pair 로 들어가. 앞부분을 index로 쓴다 ( 그냥 list 에서는 0 123을

index 로 켜는데 dict에 있는 정보를 access할 때 앞부분 a b c를 index의 수단으로 )

표제어가 str 타입으로 되었음 - int가 표제어가 될 수 있을까?

>네. ( “a” 대신 “1”을 넣어도 똑같이 apple 이 나옴 )

In [27]

a=[ (1,2,3), (3,8,0) ]

print(type (a))

print(a[0])

```
>(1,2,3)
type ( a[0])
> tuple
```

string -,abcdef

list

다시 돌아가서 f는 -1

젤 첫 번째건 늘 0 /아무리 길어도 젤 마지막건 -1 / count할 필요없음 / 끝에서 몇 번째할 때는 -

range를 해서 여러개의 정보를 가져오고 | 싶을때는 :

첫 번째에서 3번째 직전까지 [1:3]

[1:] 첫 번째에서 끝까지

[ :3] 처음에서 3번째 직전까지

list 랑 str이랑 index를 하는 방식. 정보를 가져오는 방법이 같음 = 같은 접근 방법

```
In [6]: s = 'abcdef'
        print(s[0], s[5], s[-1], s[-6])
        print(s[1:3], s[1:], s[:3], s[:])

<class 'str'>
a f f a
bc bcdef abc abcdef

In [3]: n = [100, 200, 300]
        print(n[0], n[2], n[-1], n[-3])
        print(n[1:2], n[1:], n[:2], n[:])

100 300 300 100
[200] [200, 300] [100, 200] [100, 200, 300]

In [4]: len(s)
Out[4]: 6

In [5]: s[1]+s[3]+s[4:]*10
Out[5]: 'bdefefefefefefefefef'

In [7]: s.upper()
Out[7]: 'ABCDEF'

In [10]: s = ' this is a house built this year.ㄴ'
         s
Out[10]: ' this is a house built this year.ㄴ'

In [11]: result = s.find('house')           # index of first instance of string t inside s (-1 if not found)
         result
Out[11]: 11

In [12]: result = s.find('this')            # index of last instance of string t inside s (-1 if not found)
         result
Out[12]: 1
```

len 은 변수내에 있는 정보의 길이. s 의 길이는 6 ( abcdef)

len(n) 은 총 3개

s[1] + s[3] +s[4:]

ef를 10번 곱해.

s. upper ()

>대문자로 바뀜

.점 을 하면 실행이됨.  
변수를 만들고 점 함수

s. find : s 에 담긴 string 속에서 찾아라.  
11번째에서 house 가 시작 ( 띄어쓰기 포함 )  
this 는 1번째에서 시작. / 쥬 처음 나오는 index를 찾아라 ( this 가 2개니 )

```
In [13]: result = s.rindex('this')      # like s.find(t) except it raises ValueError if not found
result

Out[13]: 23

In [14]: s = s.strip()                 # a copy of s without leading or trailing whitespace
s

Out[14]: 'this is a house built this year.'

In [15]: tokens = s.split(' ')         # split s into a list wherever a t is found (whitespace by default)
tokens

Out[15]: ['this', 'is', 'a', 'house', 'built', 'this', 'year.']

In [16]: s = ' '.join(tokens)          # combine the words of the text into a string using s as the glue
s

Out[16]: 'this is a house built this year.'

In [17]: s = s.replace('this', 'that') # replace instances of t with u inside s
s

Out[17]: 'that is a house built that year.'
```

rindex . last index 쥬 마지막 꺼의 index를 해라. 왼쪽부터 count . 여러개중 쥬 마지막 index

s.strip 잡스러운걸 지워주는. / space를 없애고, n도 없애고 순수한 텍스트만 남겨주는

s.split : s에 긴 string이 있는데, 단어 수준에서 작업하고 싶을 때. ( ' ')을 이용해서 잘라라. / s를 split 함수에 있는 입력을 이용해서 잘라라.

긴 string 을 ' '을 이용해서 잘라라. / 스페이스로 구분 .  
> ' ' 는 꼭 적고 , 로 구분 되었으면 ' , ' ??

list 로 되있는걸 복구. 스페이스를 이용해서 token에 들어있는 list들을 붙여라.  
자른걸 다시 문장으로 복구하고 싶을 때 join.  
' '을 이용해서 token에 들어있는 list들을 붙여라.  
' ' 하고 스페이스 대신 , 넣어보기.

replace / string속에 모든 this를 that 으로 바꿔라.

반복해야될 때 loop.(여러개 반복) / conditioning /

#을 붙이고 뭘 적어도 실행이 안됨/ 하나의 cell을 markdown 하면 실행이 안됨. 주석처럼 사용

```
In [2]: a = [1, 2, 3, 4]
        for i in a:
            print(i)

1
2
3
4

In [3]: a = [1, 2, 3, 4]
        for i in range(len(a)):
            print(a[i])

1
2
3
4

In [4]: a = ['red', 'green', 'blue', 'purple']
        for i in a:
            print(i)

red
green
blue
purple

In [5]: a = ['red', 'green', 'blue', 'purple']
        for i in range(len(a)):
            print(a[i])

red
green
blue
purple
```

for loop : 여러개를 여러번 해야할 때

for\_ in\_ : - 이게 문법

in뒤에 있는 것을 하나씩 돌려서 i 가 받아서 뭘갈 하라

a에 1, 2, 3, 4 / a속에 있는 걸 하나하나씩 해서 i 에 넣고 계속 다 돌아라.

첫 번째 loop가 돌아가서 i가 1을 받아 print했을 때 1이 나옴. 두 번째 loop, a의 두 번째인 2가 i 로 들어감.

in 뒤에 range라는 함수를 쓸 수도 : range뒤에 어떤 숫자가 나오면 list를 만듦. / 4가 나오면 0~3까지 - 4개의 index( 0부터 시작) 를 만듦. range가 0,1,2,3 을 만들어줬으니 loop를 돌면서 i 가 쥔 첨엔 0을 받고 , a의 1번째. a의 0 번째걸 print 해라.

그냥 print(i)를 적으면 0,1,2,3 이 나옴. - range 4라고 하면 0~3까지의 index를 만들어.

a의 길이를 적으면 / len (a) 이라고 적으면 a의 길이를. /

각각의 element를 in 앞에있는 변수에 담아둬.

range 함수를 써서 index를 만듦 ( 0, 1, 2, 3, ) 그걸 i가 받아서 a의 몇 번째가 되는.

print(i)를 하면 0,1,2,3 : range 4라고 하면 0~4

len(a) 하면 a의 길이를 적어주는 것.

첫 번째 loop에서 i는 0. range 가 0~6 . 7번의 루프를 돔.



a의 string.

for loop를 쓰지않고 print 하는 방법은

print (a[0]) ...

print (a[1])

...

for s in a :

    print(s)

a 에 있는 list를 개수만큼 for loop돌려라 - 4번. s의 variable 는 계속 바뀔.

for loop 의 내용은 indent를 반드시 해야.

a의 list를 개수만큼 for loop 에 돌려라.

range (4) or range(len(a))

print(a[s]) : a 의 몇 번째

```
In [6]: a = ["red", "green", "blue", "purple"]
b = [0.2, 0.3, 0.1, 0.4]
for i, a_ in enumerate(a):
    print("{}: {}".format(a_, b[i]*100))

red: 20.0%
green: 30.0%
blue: 10.0%
purple: 40.0%
```

```
In [7]: a = ["red", "green", "blue", "purple"]
b = [0.2, 0.3, 0.1, 0.4]
for a_, b_ in zip(a, b):
    print("{}: {}".format(a_, b_*100))

red: 20.0%
green: 30.0%
blue: 10.0%
purple: 40.0%
```

```
In [8]: a = 0
if a == 0:
    print(a)
else:
    print(a+1)

0
```

```
In [8]: for i in range(1, 3):
        for j in range(3, 5):
            print(i*j)

3
4
6
8
```

```
In [11]: for i in range(1, 3):
        for j in range(3, 5):
            if j >=4:
                print(i*j)

4
8
```

길이가 똑같은 list 두 개.

a\_ 대신 s를 씀 수업때

enumerate:

a의 list를 번호를 매긴다.

첫 번째 루프에 i 에 0. s에 red. /

그 output 값이 자기자신도 되지만 그것에 대한 번호도 매겨준다.

(변수가 두 개로 받아줌 )

i 가 번호고 s가 자기자신인 element

for loop 가 첫 번째 돌 때 i는 0. s는 red. / 두 번째는 i는 1 두 번째는 green.

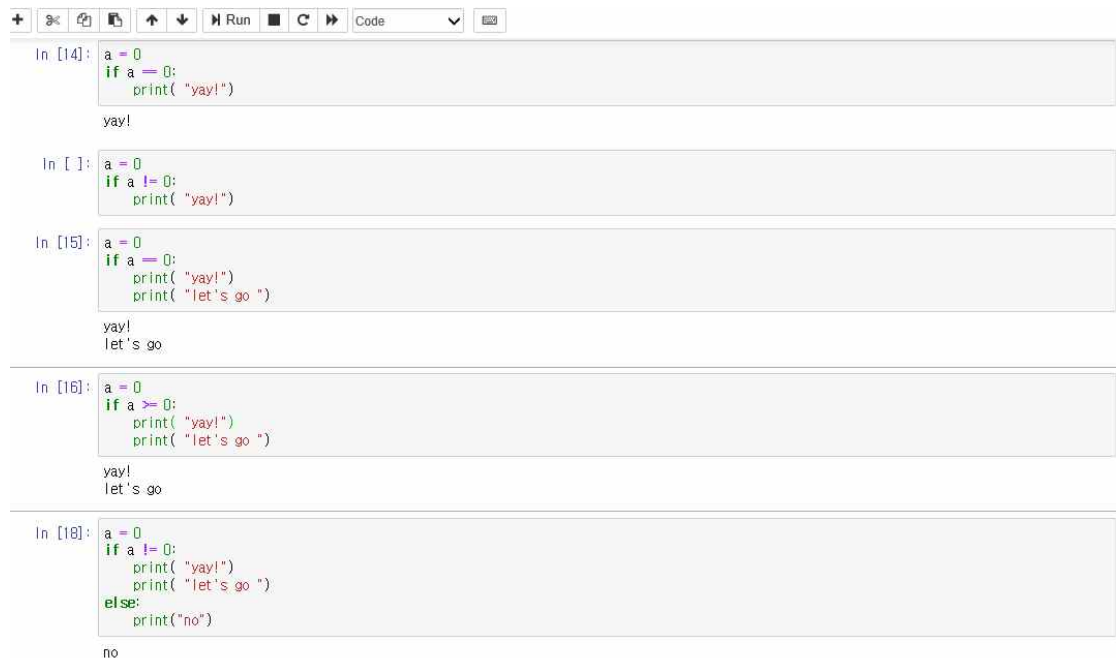
i 는 첫 번째 돌 때 0. 두 번째는 1. green

저런 형태의 format으로 적고 싶을 때 "{i} : {s} " . format에 있는 두 개가 for loop을 돌면서 중괄호속에 각각 꽂힘. s는 red. i 는 0 - b의 0 번째 : 0.2

for loop 이 4번 도는데 첫 번째 for loop 에 i,s 는 각각 0 과 red를 물고 그 다음줄로. / print 하고자 하는 format 대로. /

zip

zip을 함으로써 독립적인 4개가 pair 로 됨. loop를 돌고 첫 번째 루프에서 red, 0.2 가 s, i 로. (사진)



```
In [14]: a = 0
         if a == 0:
             print("yay!")

yay!

In [ ]: a = 0
         if a != 0:
             print("yay!")

In [15]: a = 0
         if a == 0:
             print("yay!")
             print("let's go ")

yay!
let's go

In [16]: a = 0
         if a >= 0:
             print("yay!")
             print("let's go ")

yay!
let's go

In [18]: a = 0
         if a != 0:
             print("yay!")
             print("let's go ")
         else:
             print("no")

no
```

if ~면

equal 사인을 두 개 쓰면 우리가 아는 진짜 equal sign.

a가 0 이면, colon, / ~해라 하는 부분은 indent

a가 0이 아니면 : ! / print가 안됨. / print를 여러개 해도 됨 ( indent 필수 )

> = : 0이상이면 ,

= > : 순서가 틀리면 안됨.

~하면 이거 하고, ~안하면 이거 한다 - else :

```
In [20]: for i in range(1,3) :
          print(i)
          for j in range(3,5):
              print(i*j)

3
4
6
8

In [22]: for i in range(1,3) :
          for j in range(3,5):
              if j >=4:
                  print(i*j)

4
8

In [25]: for i in range(1,3) :
          if i >=3:
              for j in range(3,5):
                  print(i*j)
```

/ 100번의 루프가 있는데 각각의 루프에 대해 50번의 루프가 돈다면 - 총 5000번이 실행.

range (1,3) : 1부터 2. (3직전까지)

젤 큰 for loop 는 2번 돈다. range 가 1,2 / 1할 때 , 2번돌고. (3,4) / 총 4번 실행

I 가 1일 때 , j loop 가 돌아감. j의 첫 번째 값은 3. 1\*3 이 print. 그 다음에 j 는 4. 원래 I는 1이었으니 1\*4. I가 2가 되면서 또 2번 되어서 총 4개의 숫자.

중간에 print(i)를 빼도됨. - 이 print는 2번 실행.

print(i\*j) 는 총 4번 실행됨.

각각의 I에 대해 j가 두 번 돈다.

3들어올때는 안됨. 총 2번 실행됨.

indent 가 안됐을때 error.

두 번 for loop 되는거 시험 !

