190905

praat
헤르츠: 1초를 44100개로 쪼개서(44100분의 1초)
숙제: 녹음하고 저장하기 WAV 파일

show pitch: 파란선 나옴, 목소리 높낮이, 여자가 높음
intensity: 세기, 노란선 나옴(폭이 넓으면 값이 높고 폭이 좁으면 값이 낮음)

wave 밑에 흑백: 소리의 spectrum(높은게 고주파, 낮은게 저주파)

190917

English consonants & vowels
철자와 소리는 다름-g/a/p

y/j/는 자음-> year라고 쓰면 자음으로 소리가 시작됨(모음으로 시작하는 ear에 자음 y가 붙은 것)

voiced/voiceless

모음: monophthongs(단모음) / diphthongs(복모음)

phonology: 소리가 어떻게 grouping되는지, 머릿속에서 일어나는 일, 머릿속의 인지 과정, study on sound 'system'
phonetics: 더 물리학적, 늘 차이가 생김, physical(가를 똑같이 열 번 얘기한다고 머릿속으로는 생각하지만, 실제로는 미묘한 차이가 생겨남), study on speech
speech: 사람이 하는 모든 말

'아'에서 '이'로 소리가 바뀌는 이유? 입모양 때문(입모양 속에 혀의 위치, 턱 등 포함-주된 요인은 아님)

한국어는 음절이 반복, 영어는 stress가 반복
한국어는 턱을 많이 쓰는 언어, 영어는 혀를 많이 쓰는 언어

articulatory: 말의 시작, 소리를 만들어내는 사람의 원리
acoustic: 공기와 소리가 어떻게 조화되는가, 사람과 관계없이 완전 물리적 특성
auditory: 어떻게 듣는가, 사람이 수반된 매커니즘, 고막이 움직이는 것 고려(물리학)

성대(larynx): 후두

인강(pharynx): 목젖부터 후두까지 긴 관
구개(palate): hard / soft(velum)
alveolar: 아주 중요한 부분
목젖(uvula)

upper structure는 그대로 있고 lower structure가 움직임(e.g. lip, tongue tip, tongue, epiglottis- 기도로 가는 길을 막음)
말할 때 식도는 이용하지 않고 기도를 이용

입말고 코로 가는 tract도 있음

비음을 뺀 모든 자음(nasal)은 velum이 raised된 상태에서 소리가 남(nasal은 lowered)
코로 숨을 쉴 때: velum은 lowered된 상태

larynx: 막히면 진동, 유성음/무성음 나눔

phonation process in larynx

articulation: 성대에서 유성/무성 구분, velum에서 lowered/raised되는 것

190919

유튜브에 남호성 교수님 강의 올라옴

조음, 음향(acoustic): 말하는게 어떻게 공기를 타고 가는가, 청각(auditory): 우리 뒤로 들어가는 과정

조음: 크게 세 가지 요소(ppt에 세 동그라미)
제일 중요한 부분: 혀를 중심으로

숨을 쉴 때: velum은 내려옴-> 코로 가는 길이 생김
velum이 lowered 되면 m. n, ng 등 소리가 남

lip, tongue tip, tounge body 모두 constrictor(협착을 만드는 주체)
constriction location: 앞뒤 조절
constriction degree: 상하 조절
constriction location과 constriction degree에 의해 자세히 조정됨

lip이 앞으로 가면 b, 뒤로 오면 v
tongue body가 앞으로 오면 j

tongue tip: 윗니(th), alveolar(d, t, n), ...
constriction location 관점에서 lip, tongue body는 2개, tongue tip은 네 개 정도로 미세 조정됨

constriction degree: stops, fricatives, approximants(영어에서 네 개- r, l, w, j), vowels

!!시험
velum raised, glottis(larynx의 틈) open, constrictor tongue tip, CL alveolar, CD stop 소리는? /t/
모든 모음은 constrictor로서 tongue body만 씀(tongue tip과 lips는 안 씀)
모음과 같은 constrictor를 쓰는 자음의 예시 중 velum lowered? /ng/- glottis closed(진동 일어남-유성음)

phoneme: 개별적인 소리
모음과 자음 specify하는거 시험

소리를 measure하는 방법
duration: 길이, 자음은 짧고 모음은 길
pitch: 파란 곡선, 높다가 낮아지는 그래프(ppt), 소리의 높낮이(Hz), 여성의 목소리인지 남성의 목소리인지에 따라 setting-range를 다르게 해줘야함
intensity: 소리가 큰지 작은지 (같은 도라도 약하거나 세게 말할 수 있음)(dB)
spectrum: 소리가 1차원적인 것이 아님, frequency 관점에서 분석

spectrogram: 프리즘으로 빛을 분산시키듯이 frequency 관점에서 분석하는 것
까만 띠가 보임- formant라고 함
formant: 흑백의 띠, 모음이 뭔지 결정, 모음을 구별하는 수치적인 지표- formant 값에 따라 이 모음은 뭐라고 말할 수 있음, 줄여서 f1, f2 이런 식으로 말함

190924

어떻게 자음과 모음을 발성하는가?
세 가지 중요한 과정- larynx, velum, 입
p: lips-> location- bilabial, degree- stop, velum raised, larynx open
b: p에서 larynx만 closed로 바뀌면 됨
d: tongue tip-> location- alveolar, degree- stop
z: tongue tip-> location- alveolar, degree- fricative
n: tongue tip-> location- alveolar, degree- stop, velum lowered, larynx closed

praat 시험에 나올 수 있음

Hz: measure의 단위, 1초 동안 크게 반복되는 것이 몇 번 나오는가 (the number of occurrences of a repeating event per second)
1초에 sine wave가 몇 번 나오느냐에 따라 주파수를 얘기할 수 있음
frequency: 1초에 몇 번 반복되는지
magnitude: sine wave의 크기
vocal fold의 vibration에 의해 repeat이 일어남

sine wave를 결정짓는 것은 frequency와 magnitude(얼마나 소리가 큰지)
이 세상에 존재하는 모든 signal(sound 포함)은 여러 다르게 생긴 sine wave의 결합으로 표현됨
모든 신호는 조금씩 다른 sine wave의 합

파란 그래프: frequency가 상대적으로 높음(첫 번째 빨간 것보다 3배 빠름)
frequency가 작음-> slow하다는 것, 저음
magnitude는 두 번째 초록색이 제일 작고, 첫 번째 빨간색이 제일 큼
여러 sine wave의 합은 sine wave가 아닌 복잡한 소리로 만들어짐
복잡한 소리, 혹은 신호는 다양한 sine wave의 합으로 표현됨
합(complex tone): 1초에 100번 반복됨(빨간 wave와 똑같은 반복되는 주기)
sine wave에서 x축: 시간, y축: 단순한 숫자값, value(voltage)
time-value의 그래프를 frequency-amplitude 그래프(spectrum)로 변환시킬 줄 알아야함

spectrum이 어떻게 이루어져있는가? 우리 주변에서 보는 소리는 복잡한 형태
spectrogram- spectrum을 time으로 visualize함(spectrum은 시간개념이 없음, 한 given point에서 어떤 frequency 성분이 많은가를 보는 것), spectrum을 시간축으로 늘려놓은 것

아 라고 녹음하고 spectrum slice를 확인했더니 130Hz 간격으로 반복됨- 등간격 (아까 pure tone을 만들었을 때는 440hz하나만 있었는데, 사람 목소리는 complex tone이기 때문에 여러 개가 생김)
맨 처음 나오는 130Hz는 목소리의 pitch와 일치- 제일 낮은 주파수의 것과 일치(ppt- 100Hz의 반복 패턴이 합성한 그래프에서 나타나는 것처럼)
-> 사람의 음의 높낮이는 어떤 frequency와 일치하는가? 제일 작은 pure tone(sine wave)의 진동수와 일치(위에서는 130Hz)
-> 아 라고 얘기를 할 때, 여러 다른 simplex tone의 합으로 녹음됨-> 제일 slow한 simplex tone의 frequency가 우리 목소리의 pitch(음높이)와 동일-> 성대에서 vocal folds가 1초에 몇 번 떨리는지와 일치

진동수- frequency, 1초에 몇 번 반복되는지, 단위는 Hz

source(성대에서 나는 소리)에서 filter(tube, 입모양 등)가 어떻게 바뀌는지에 따라 ㅏ 소리도, ㅣ 소리도 만들 수 있음

graph가 decreasing- 모든 사람의 source의 패턴
제일 처음에 나오는 F0(fundamental frequency, pitch, the number of vocal fold's vibration in a second)이 이후 곱하기되어서 나옴(115->230->...)- harmonics(배음- 배를 이룸)
speech의 source는 sine wave의 합처럼 저렇게 생김(ppt에서 gradually decreasing하는 그래프), 점점 amplitude가 작아짐
모든 사람의 source는 똑같은 패턴이지만, 거리는 다를 수 있음- 여자면 첫 시작이 남자보다 높을 것-> graph 모양이 더 듬성듬성인 모양일 것 (10000Hz에서 자른다고 생각했을 때, 남자가 갖는 배음의 숫자가 여자보다 많음- 시험)

head 소리의 graph- 배음의 구조가 안 깨짐(115의 배수대로 가는 건 그대로 유지)
amplitude가 깨짐- smoothly decreased가 아닌 제멋대로 왔다갔다

spectrogram은 wave와 쌍을 이룸
x축은 wave와 마찬가지로 time이지만 y축은 frequency
밑은 까맣고 위로 갈수록 옅어짐, 진한 부분이 amplitude가 큼
low frequency 쪽에서 energy가 크고, high frequency에서 에너지가 약해짐 (까맣게 생긴 것이 에너지가 큼)-> 옆에 analysis한 graph도 마찬가지

190926

모든 소리: simplex sound의 합
sine wave: 다양하게 패턴 바꿈
wave form: x축이 시간, y축은 그냥 value

spectrogram에서 x축은 시간, y축은 frequency(!시험)

sine wave에 F0가 정해지고, 그것의 배음의 합으로 목소리의 source가 정해짐
목소리의 pitch는 첫 번째 frequency와 일치- F0 (단위는 Hz)
저주파에선 에너지가 높고, 고주파로 가면 약해짐

입모양이 filter 역할
harmonics가 gradually decreased되지 않음
위의 source와 마찬가지로 harmonics가 일정 간격 유지

source: peak가 없음, gradually 낮아지기 때문-> 산맥 형성 안 됨
pure tone들이 high frequency로 갈수록 decrease되는 것이 voice source의 패턴

누가 /아/라고 말하든 산맥은 똑같은 패턴으로 나타남
산맥- formants

harmonics에서 제일 첫 번째- F0(pitch)
산맥에서 첫 번째, 두 번째.. 산맥을 f1, f2... 라고 말함(harmonics와는 다름)

fundamental frequency: 220Hz (guitar)
기타는 complex tone- 사람의 목소리와 똑같음 (220, 440... 의 합으로 되어있음)

100Hz부터 1000Hz까지 combined mono된 소리: 반복주기는 F0와 일치- 인지심리학적으로 100Hz(F0)의 높이와 비슷하다고 느껴짐

F0: 첫 번째 나오는 것의 frequency(Hz)
F1: 처음 나오는 peak
F2: 두 번째로 나오는 peak

영어와 한국어의 /아/: 영어가 더 back, low한 소리

191001

컴퓨터 언어는 여러 종류가 있음(java, python 등)
다 다르지만 공통점 있음- 모든 language는 단어(정보가 들어 있음, 정보를 담는 그릇)와 문법(단어를 어떻게 combine?)이 존재

변수(variable): 단어에 해당, 정보를 담는 그릇의 역할 (정보의 종류는 많지 않음- 숫자와 글자 두 가지)
기계의 문법은 생각만큼 어렵지 않음
컴퓨터 문법: 변수라고 하는 그릇에 정보를 담음, conditioning(if문법), 여러 번 반복(for-loop?), 함수를 배움(입력-출력)(제일 중요)

디렉토리: 컴퓨터에서 어디에 위치해있는지

=: 오른쪽에 있는 정보를 왼쪽의 variable로 assemble하는 것 의미 (1이라는 정보를 a라는 변수에 넣음)

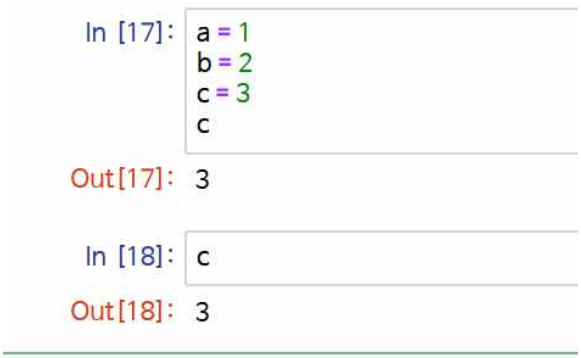
python의 모든 함수는 누가 만들어놨던걸 수도 있고, 내가 만들 수도 있고

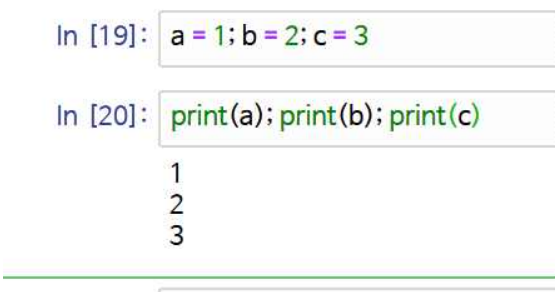
print라는 함수의 입력: a
입력을 표시해주는 것: 괄호()

셀을 초록색에서 파란색으로 바꾼 후 b라고 치면 below에 셀을 만들어줌
x를 치면 없어짐
print(a)라고 쳤을 때 1이 나오는 것은 새로운 셀이 아니라 결과값
a=2라고 치고 run을 눌러 실행하면 1은 없어지고 2가 됨-> print(a)를 했을 때 2가 나옴
print(a)=1인걸 실행하고 밑에 print(a)를 실행하면 2가 아니라 다시 1이 됨

문자를 입력하고 싶을 땐 반드시 quote를 넣어줘야 함(e.g. b = 'love')
run의 단축키: shift+enter

love = 2
b = love (b = 'love'였던 아까와는 다름)
라고 한 후 print(b)=> 2라는 결과값이 나옴

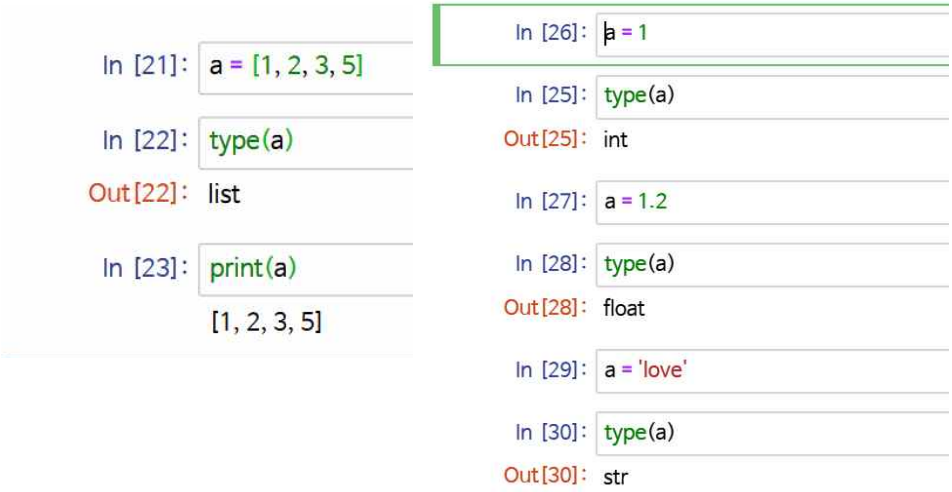




여러 번 입력한 것 중 변수를 한 번 더 써주면 print 함수 없이도 그 결과값을 보여줌, 다음

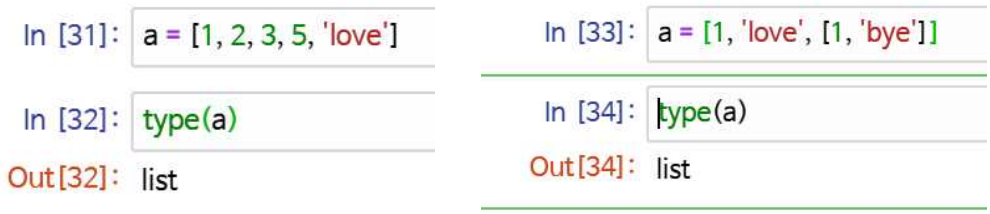
셀에 c를 그냥 쳐도 3을 보여줌

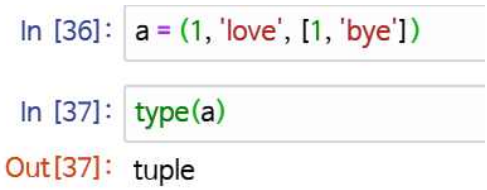
variable은 quote없이, 문자 정보는 반드시 quote(single, double 상관 없음)



[]: 여러 숫자를 한 번에 넣을 수 있음(e.g. a = [1, 2, 3, 5])

type의 종류: int, float, str, list, tuple, dict...
(type 종류 묻는 거 시험에 나올 수 있음)





list- 반드시 숫자일 필요는 없음(e.g. a = [1, 2, 3, 5, 'love'])
list와 tuple은 완전 똑같음- list는 대괄호 사용, tuple은 그냥 괄호 사용
tuple이 보안에 더 강함

a = {'a': 'apple', 'b': 'banana'} - dictionary에 2개를 넣은 것 (콤마로 인해 2개가 들어와 있음, 중괄호를 씀- 중괄호를 써야 dictionary, 몇 개를 넣을지는 콤마로 표현, 설명의 쌍은

콜론으로 표현, 콜론 안에 string만 들어갈 필요는 없음)

191008

중간- 오픈북, 암기 필요 없음, 이해해야하는 문제 위주로
순서: variables-> string-> syntax-> numpy (sound는 중간고사 이후)

```
In [6]: a = {"a":"apple", "b":"orange", "c":2014}
        print(type(a))
        print(a["a"])

        <class 'dict'>
        apple

In [9]: a = 1
        b = 1
        c = a+1
        print(c)

        2
```

제일 advanced된 것- dict
list는 단편적 정보를 넣지만, dict는 페어(콜론)로 넣음(두 개짜리 페어가 여러 개- a-apple, b-orange, c-2014)

```
In [13]: a= [1, 2]
         b = [3, 4]
         c = a[0]+b[0]
         c

Out[13]: 4

In [15]: a= [1, 2]
         b = [3, 4]
         c = a[1]+b[1]
         c

Out[15]: 6
```

a 리스트에서 0번째, b에서 0번째를 가지고 와서 더하면 4
각각 1번째를 가지고 와서 더하면 6

```
In [16]: a = 1.2; a = int(a); print(type(a))
         <class 'int'>
```

```
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'>
```

세미콜론: 두 줄 적을거 한 줄에 적고 싶을 때
float 함수: 어떤 variable이 들어오면 그것을 float로 바꿔줌(e.g. 원래 int였던 a=1이 float로 됨)
int 함수: int로 바꿔줌 (e.g. float를 int로 바꿔줌) a = 1.2; a = int(a); print(type(a))=> int

```
In [18]: a="123"; print(type(a)); print(a[1])
<class 'str'>
2
```

어떤 variable의 내부 정보에 들어갈 때는 반드시 대괄호를 씀
대괄호 안에 들어가는 것은 index
variable 이름을 a라고 적고-> 대괄호 열고 닫고-> 내부적 정보를 부분적으로 가져옴
-> 2라는 문자를 가지고 옴(숫자가 아님)

```
In [17]: a=123; print(type(a)); print(a[1])
<class 'int'>
---------------------------------------------------------------
TypeError                 Traceback (most recent call last)
<ipython-input-17-b0076a8d2193> in <module>
----> 1 a=123; print(type(a)); print(a[1])

TypeError: 'int' object is not subscriptable
```

a='123'이 아닌 a=123을 넣었을 때(quote를 뺐을 때) 에러 나는 이유: 0번째 것밖에 없기 때문

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

str(문자)이었던 것을 list로 바꿈-> 123이 쪼개져서 각각 문자로 list화 됨

```
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']
```

1은 숫자로서, 2는 문자로서 print된 것임

```
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 [23]: a=[(1,2,3), (3,8,0)]
         print(type(a))
         print(a[0])
         type(a[0])

         <class 'list'>
         (1, 2, 3)

Out[23]: tuple
```

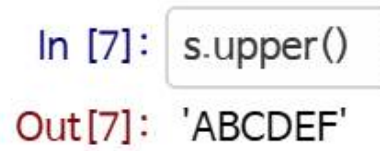
24번
a, b, c: index로서 기능 (페어에서 앞부분이 index)
사전과 같음- 표제어(a, b, c)와 내용
index의 type: 전부 다 string(quote로 되어 있음)
“a”를 1로 바꿔도 똑같음(quote 빼도 똑같음)

```
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
```

제일 앞은 늘 0, 제일 마지막은 늘 -1, 끝에서 몇 번째를 구할때는 마이너스 기호를 붙임
[1:3] - 첫 번째부터 세 번째 직전(=두 번째)까지 가져온다는 뜻-> bc가 됨
[1:] - 첫 번째부터 끝까지
[:3] - 맨 처음부터 두 번째 까지
[:] - 전부 다

list(3번)와 string(6번)은 똑같은 방식으로 정보를 가져옴
print- 콤마를 찍으면 한 줄에서도 여러 개를 print out 할 수 있음

len: variable 내에 있는 정보의 길이, 개수
s의 길이: 6(abcdef)
n의 길이: 3



upper: 일종의 함수, s 속의 문자가 대문자로 바뀜, s를 upper 시킨 함수
파이썬과 같은 language에서는, 어떤 variable을 만들고 그 옆에 .을 붙여 함수같이 쓰면 실행됨

find: 찾아주는 함수 (string-7: s 속에 담겨있는 string 속에서 무언가를 찾는 것)
결과: 몇 번째로 나오는지 (e.g. 11번째에서 house가 시작됨- 제일 앞 띄어쓰기(space) 때문에 11번째가 됨 / this- 두 번 나오는 것 중 제일 처음 나오는 것을 찾아줌)
rindex: 여러 개 중 제일 마지막 index
strip: 잡스러운걸 지워주는 함수, 순수한 텍스트만 남겨줌

```
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.'
```

```
In [17]: s=','.join(token)
         s
Out[17]: 'this,is,a,house,built,this,year.'
```

split: . 앞에 오는 긴 string을(e.g. s) split 함수에 있는 입력을 이용해서 자르라는 뜻, 단어 수준에서 작업하고 싶을 때, 긴 string을 자름(여기서는 ‘’안이 space이므로, space를 기준으로 자름, 콤마가 사용되었으면 콤마일 수도 있음)
tokens: 임의로 정한 variable의 이름
join: split을 이용해 잘라 list로 자른 후 이것을 다시 문장으로 복구하고 싶을 때 사용
. 앞에 space 입력-> space(‘ ’)를 이용해 token에 들어있는 list를 붙이라는 뜻
replace: e.g. 이 string 속의 모든 this를 that으로 바꿔라

191010

syntax 문법
loop: 여러 개를 반복할 때 씀
if: 조건, 컨디셔닝
function: 입력-출력

셀에 comment다는 방법: # 다음 글씨를 씀
markdown: 셀에 무엇을 써도 실행이 안 됨, 주석의 기능 (code라고 적힌 버튼을 누르고 markdown으로 바꾸면 됨)

for loop: 여러 개를 여러 번 할 때
for: 어떤 language든 똑같이 씀

```
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(4):
          print(a[i])

1
2
3
4
```

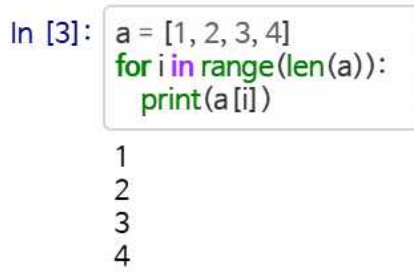
for i in a:
in 뒤에 있는 것(a)을 하나씩 돌려서 i가 그 하나하나씩을 받아 무언가를 하라를 콜론 밑에 적어주면 됨
a 속에 있는 것을 처음부터 하나씩 i에다가 넣어라(할당해라)
1번째 루프- i가 1을 받아 print하면 1이 나옴, 2번째 루프- 2가, 3번째는 3이, 그 다음에는 4

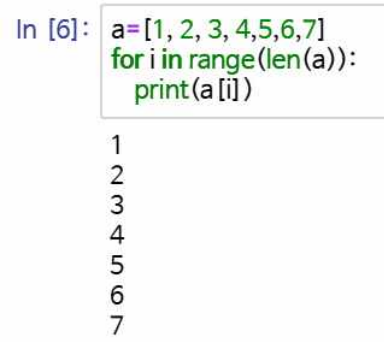
in 뒤에 list를 그대로 쓸 수도 있지만 range함수를 쓸 수도 있음
range 뒤에 어떤 숫자가 나오면 list를 만들어줌- e.g. 4를 넣으면 0부터 3까지 list를 만들어줌- 4개의 index를 만들어줌(0부터 3까지)(index는 0부터 시작)
-> 루프를 돌면서 a의 i번째가 print됨, 제일 처음에는 0번째 것(1)(i에 0이 들어와 있기 때문)-> a의 0번째 것이 print됨)이 print되고 쭉 이어지는 식

for loop 쓰는 두 가지 방법- 리스트를 그대로 쓰거나(각각의 element를 in 앞의 variable에 담아둠), range 함수를 씀(0부터 얼마까지 index를 만들어주고 그것을 i가 받음)

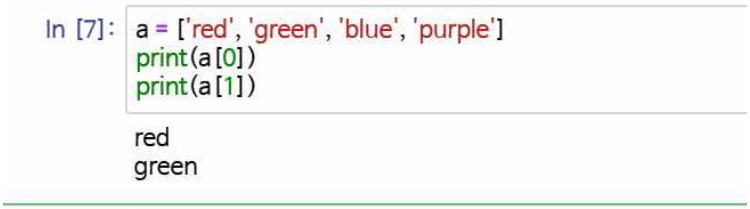
```
In [4]: a=[1, 2, 3, 4]
        for i in range(4):
          print(i)

0
1
2
3
```

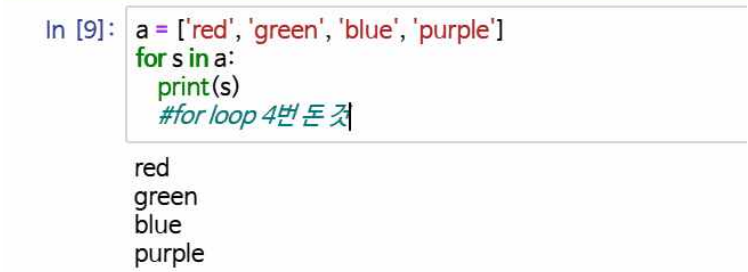
앞의 range 함수 예시에서 그냥 print(i)라고 하면 0,1,2,3이 나옴- range(4)는 0부터 3까지 4개의 index이기 때문





in[6]
range에 len(a)를 넣으면- 7개의 index(0부터 6)가 만들어짐. 그것이 for loop를 돌면서 i 받음
제일 첫 번째 루프에서 i는? 0 (range는 0에서 6)
-> a의 0번째, 1번째.. 이런 식으로 가게 됨





for loop를 쓰지 않고 결과를 내는 방법: print(a[0]), print(a[1]) 이런 식으로 노가다
for loop의 내용은 반드시 indent가 있어야 함(for loop를 치고 콜론 치고 엔터하면 자동으로 indent 생김)
in[9]
a list를 개수만큼 for loop를 돌려라- for loop를 4번 돈 것
s variable은 4번 바뀜

```
In [13]: a = ['red', 'green', 'blue', 'purple']
         for s in range(len(a)):
           print(s)

0
1
2
3
```

```
In [14]: a = ['red', 'green', 'blue', 'purple']
         for s in range(len(a)):
           print(a[s])

red
green
blue
purple
```

in[13, 14]

a의 길이만큼 range를 만들어서 loop를 돌리라는 뜻의 함수

13- 0, 1, 2, 3이 나옴-> range의 index 값이 4개가 만들어지기 때문(0부터 3)

14- a의 몇 번째를 print

```
In [15]: a = ['red', 'green', 'blue', 'purple']
         b = [0.2, 0.3, 0.1, 0.4]

         for i, s in enumerate(a):
           print(a[i])

red
green
blue
purple
```

```
In [16]: a = ['red', 'green', 'blue', 'purple']
         b = [0.2, 0.3, 0.1, 0.4]

         for i, s in enumerate(a):
           print("{}: {}%" . format(s, b[i]*100))

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

in[15]

a, b 각각 string과 number가 있는 길이가 같은 list

enumerate 함수: ( )안 list의 번호를 매김, output 값이 자기 자신도 되지만 그것에 대한 번호도 매겨줌 e.g. a의 list의 번호를 매김

variable이 두 개가 들어가 있음(i, s)- 번호까지 받아줘야 하기 때문(앞이 번호, 뒤에가 자기 자신인 element)

i는 번호가 enumerate되서 들어온 것-> a의 몇 번째 이런 식으로 4번 loop (s를 안 씀)

in[16]

첫 번째 variable i에는 index(번호값)를 받아오고, 두 번째 variable s에는 list의 값을 받음

-> 첫 번째 루프가 돌 때 i는 0, s는 red가 들어옴, 두 번째는 i는 1, s는 green 이런 식으로 들어옴
double quote를 하고 내가 원하는 variable의 개수만큼 중괄호를 써줌(format 안 2개가 for loop를 돌며 중괄호 속에 꽂힘)- format 안 콤마로 연결된 것의 개수랑 quote 속 중괄호 개수가 똑같음(2개)
format 함수 속에 두 개가 들어있음- 그 두 개가 for loop를 4번 돌면서 매번 두 개의 중괄호 안에 들어감
red와 20(b의 0번째인 0.2에 100을 곱한 값)이 double quote안의 중괄호 형태 속에 들어감
b[i]: b의 i번째 것

```
In [18]: a = ['red', 'green', 'blue', 'purple']
         b = [0.2, 0.3, 0.1, 0.4]

         for s, i in zip(a, b):
           print("{}: {}%" . format(s, i*100))
```

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

a, b는 반드시 같은 길이의 list
zip: 두 개를 세트로 합침(독립적으로 존재하는 두 개의 list가 페어가 됨-> 루프를 돔)
첫 번째 루프에서 red와 0.2가 각각 s, i에 들어감

```
In [24]: a=0
         if a == 0:
           print("yay!")
           print("let's go")
```

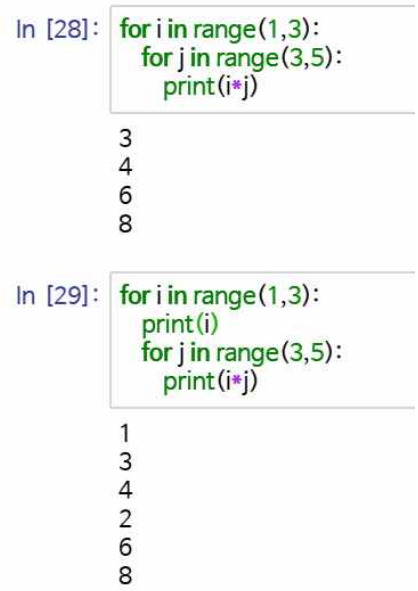
```
yay!
let's go
```

```
In [27]: a=0
         if a != 0:
           print("yay!")
           print("let's go")
         else:
           print("no")
```

```
no
```

만약 a가 0이면 yay!와 let's go를 print 해라
equal sign을 두 개 넣어줌(==)-> 진짜 equal sign이 됨 (if에서 equal sign을 많이 씀)
>=: 크거나 같음(부등호 순서 바뀌면 안 됨- equal sign 앞에, ==대신 넣어줌)
!=: 아닌 경우 (if a !=0: print("t") - a가 0이 아니면 t를 print 해라)
else: 맞으면 ~하고, 아니면 ~하고

range 다음 숫자 하나가 들어가면? (e.g. range(10)- 0부터 9까지 index가 만들어짐)
숫자 두 개가 들어가면?
e.g. for i in range (1,3):
print (i)
<결과>1, 2
: 1부터 3 직전까지 감-> 1, 2가 i에 받아지면서 loop가 돎



in[28]
시험- 두 번 for loop
제일 바깥 for loop- 2번 돎(range가 1, 2 이렇게 돎)-> 각각 1, 2 for loop 할 때도 2번씩 돎
-> 총 4번 실행됨
-> 1x3, 1x4, 2x3, 2x4인 결과가 나옴

in[29]
print(i): 첫 번째 for에서만 실행됨, 2번 실행됨
print(i*j): 4번 실행됨

```
In [30]: for i in range(1,3):
             for j in range(3,5):
                 if j>=4:
                     print(i*j)

4
8
```

```
In [31]: for i in range(1,3):
             if i >= 3:
             for j in range(3,5):
                 print(i*j)
           File "<ipython-input-31-cd2f4c498173>", line 3
             for j in range(3,5):
             ^
         IndentationError: expected an indented block
```

in[30]
일단 크게 for loop- i가 바뀌면서 1부터 2까지 2번 돎-> 각각의 i에 대해 j가 2번 돎
실행에 if를 걺- j가 4보다 커야한다는 조건을 통과해야함(j가 3일 때는 안 돌 것이고, 4일 때만 돌 것)
-> 4번이 아니라 2번만 돎

in[31]
error: if 밑에 나오는 for가 indent되어야 함
indent 되면? 아무런 결과가 안 나옴- i가 1, 2 이렇게 돌아 3보다 크거나 같을 수 없기 때문

191015

집중적으로 배웠던 부분: articulation
5 speech organs 그림 중요(velum, major articulation 3가지(lips, tongue tip, tongue body)- 제일 중요, larynx), 어떻게 소리 specify?
larynx: vocal cord가 vibration-> 유성음/무성음
velum(soft palate): lowered/raised (nasal sound를 나눔)
lip, tongue tip, tongue body

constriction degree: 혀를 어느 정도로 올리는지
constriction location: 혀가 앞으로 가는지/ 뒤로 가는지

constriction degree: approximants(r, l, j, w)
모든 영어 phoneme은 stop, fricative, approximant, vowel 중 하나

velum, larynx, lip/tongue tip/tongue body, constriction degree and location- 모든 가능성 중 거기 해당하는 영어 phoneme이 없을 수도 있음- 어떤 조건인지 생각해올 것
lips가 constriction location으로 velar 가질 수 있나? 영어에서의 gap인가 사람이 못 하는

건가? 사람이 할 수 없는 소리
lips가 alveolar를 constriction location으로 가지는 language가 이론상 가능한가? 가능하긴 함(accidental gap), velar는 physical하게 불가능

praat: pitch, intensity, formant가 뭔지
중요! source-filter theory 그림을 설명할 수 있어야 함
source 부분에서 pitch가 조절됨, pitch는 제일 처음에 있는 pure tone에 의해 결정됨(F0(pure tone의 frequency)의 크기가 pitch)
입모양이 filter 역할-> source spectrum이 filter에 의해 재구성됨(peak와 valley가 결정됨, source에선 peak 없음, 제일 첫 번째에 있는 peak- first formant(f1))
F0과 f1, f2는 다름- 그림도 다름
f1과 f2에 의해 모음이 결정됨
f1: 높낮이 결정
f2: front, back 결정, f2가 클수록 front(i>e)

variable- number(int/float)/str
하나 이상의 정보 저장- list(대괄호), tuple(괄호)
페어로 묶음- dict(표제어와 내용)

시험! str과 list는 유사
list는 index 이용(정보에 부분적으로 access)
str도 정보 access 할 때 대괄호 사용할 수 있음- list와 똑같음

dict도 콜론의 앞부분을 index로 이용할 수 있음

제일 끝: -1, 그거에서 하나 앞: -2
range 이용하고 싶을 때: 콜론 사용

split이랑 join이 중요
split: s속에 담긴 str을 split 다음에 있는 캐릭터를 가지고 spilt하라

함수를 잘 알아야 함: print, range, len 등

indent 다음 내용이 for이 지배하는 부분
for in: in 뒤에거부터 먼저 읽음, in에 있는 것들을 하나하나씩 루프를 돌려서 in 앞에 있는 거에 넣음

enumerate, zip, format 어떻게 쓰는지 잘 보기

if 쓰는 방법: 콜론 꼭 써야함(else를 쓸 때도)

모음: 무조건 tongue body

int, float, join, split 이런 것도 함수임

파찰음은 안함

여자가 남자보다 F0가 높음
듬성듬성 있는 것이 여자 것(F0가 높기 때문)- 주어진 frequency range 속에서 남자의 pure tone이 더 많음
e.g. 남자가 105Hz면 0부터 10000까지 몇 개 들었는가? 105, 210, ... 세보면 됨 (filtered되어도 숫자값은 안 변함)

191029

영어의 nasal consonant는 velum lowered-> nasal tract으로 air flow가 있지만 oral tract는 막혀서 air flow가 없다: T
영어모음 /a/, nasal tract의 air flow 차단됨: T
발화시 air pressure와 가장 관계있는 것: 진폭과 관계가 있음-> intensity
자음 중 코를 막고 숨을 쉴 때 가장 유사한 articulation 상태인 음소? h
영어 /h/는 어떤 articulator(lip, tongue tip, body)에서 constriction? none
음소 중 tongue body에서 constriction, constriction degree는 frecative, location은 velar인 자음 개수: 없음
영어 자음 /v/의 articulator의 location을 bilabial, degree를 approximant로 바꾸면 어떤 음소? w
/s/, /l/은 major articulators 중 공통적으로 tongue tip에서 조음됨
어떤 모음의 pitch가 128Hz일 때 F1은 128Hz보다 반드시 크다: T
/i/ 모음 128Hz로 발화, 똑같은 pitch로 /a/ 발화, 각각의 모음에 대해 0Hz~5000Hz 사이 몇 개의 harmonics 존재하는지 구하면? 같은 개수

```
a=[[1], [2,3], [4,5,6]]
n=0
for b in a:
 for d in b:
    n+=1
print(n)
```

6(1번 돌고, 2번 돌고, 3번 도니까)

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

몇 개의 unique한 함수가 있는가? range, len, print 3개
함수가 몇 번 실행되는가? len(1번), range(1번), print(4번 실행)- 6번

a=[1,2,[3,4]]
print(a[-1][-1])
4
이유: [3,4]-> 4

b= [1, ‘a’, {‘a’: [3,6], ‘b’:[9]}, ‘b’]
print(b[-2][‘b’][-1])
9

c=[1, ‘a’, {‘a’: ‘abc’
print(c[-1][‘b’][-2])
e

print(‘join(d)[-3]
h

17번: e에 list 4개-> for loop 4번 돎
dict라고 나오는 것이 없음
k만 print out됨

18번: range 3: 0,1,2
14*1

19번: a의 length는 4

20번: tuple, element 3개
len(a): 3
int(10): 10

for loop는 함수가 아님, 함수는 반드시 함수명과 괄호가 있어야함

행렬 알아야함- 직사각형 안에 숫자
vector화 해야함
이미지는 행렬임, 그것을 길게 늘어놓은 것을 vector화 한다고 함
모든 데이터는 vector의 형태로 해야 다루기 쉬움
컬러: 행렬이 3겹으로 있음(RGB)
흑백 이미지: 2차원, color image: 3차원, 시간까지 있으면 4차원
이미지도 벡터, 소리도 벡터화 됨, 텍스트도 벡터화 됨-> 모든 데이터는 벡터화

library: 쓰기 좋게 함수들을 누가 만들어놓는 것, 추가해서 우리가 쓸 때 library를 불러옴

numpy: list 중에서도 그 안에 숫자가 들어갈 때

in[3]: list 안에서는 수학적 계산이 일어나지 않음, 수학적 계산을 해주는 것이 numpy
import numpy를 해와야 numpy를 쓸 수 있음
array: list를 행렬 혹은 벡터로 만들어줌, 계산이 가능하게 바꿔줌(list에선 계산 불가)
numpy를 np로 많이 줄여씀

X.shape: 차원(2 by 3)- 2행 3열의 matrix(행렬)

191031

numpy를 import해야 함
numpy라는 library 속에 다른 package들이 있을 수 있음
from numpy로 쓸 수도 있음- from numpy import A(numpy 안에 있는 A를 불러오자), from numpy import A.D

matplot: plotting과 관련됨
import matplotlib.pyplot=from matplot
.은 library의 포함관계를 말함

numpy: list와 비슷
수학적으로 계산할 수 있기 때문에 사용-> list가 아닌 numpy 처리를 해야 함

empty: 함수(괄호로 input 받음)

in[2]
리스트가 두 개 나옴, 안에 있는 숫자는 랜덤, int라고 적어뒀기 때문에 소수점은 아님, 여러 번 실행할 때마다 값이 달라짐

in[4]
0으로 채워져서 나옴

in[5],[6]
2 by 3(2행 3열)- 리스트(아무 쓸모 없음)
이 리스트를 array로 바꿔줌
[5]와 [6] 같음- list를 array로 convert한 것

in[7]

1이 만들어짐
.이 있음- int가 아니라 float(default로 data type을 float로 함)

arange, linspace- 중요한 함수

in[9]
계산이 되는 array를 만들어줌- 숫자 5개가 나옴(index 5개, 0부터 4까지)
in[10]
0부터 10 전까지 index 만들어줌
in[11]
0부터 2씩 뛰면서 만들어줌

in[12]
linspace: 각 값의 space가 똑같음(각 값의 차이가 똑같음)

in[13]
3 by 2의 벡터
처음과 끝 대괄호 2개가 붙어있음-> 2차원 (3개가 붙어있으면 3차원)
in[14]: 3차원

X.ndim: 3차원
X.shape: 3x2 직사각형이 2개 있음

X.dtype: 원래 type

in[20]: 숫자를 다 0으로 바꿔줌 (X*0만 해줘도 됨)

normal: normal distribution을 만들어주는 함수
괄호 안에 차례로 mean(0), standard deviation(1), data의 개수(100)
in[21]
100개의 랜덤한 데이터가 나옴
data ndim하면 1차원, shape은 100

histogram
y값 절대 소수로 나올 수 없음, 전부 정수값
y축 합하면 몇 개인지 시험에 나올 수 있음

in[25]
2 by 3 by 4
안에 있는 element의 개수: 24

reshape: shape을 바꿈
element 개수는 안 바뀜
-1: 원래 4를 적어야하는데, 모르겠을 때 알아서 하란 의미로 앞에 -1이라고 침
element는 여전히 24개

allclose: 두 함수가 똑같은 형태인지 비교
assert 몰라도 됨

randint: 0부터 10 사이 숫자를 pick up해 2 by 3의 matrix로 만들어줌
random: random을 2 by 3로 만들어냄
savez: 실제 file로 저장해줌

data를 메모리에서 지우고 싶을 때: del하고 variable 이름을 적음

who: 몰라도 됨(available한게 뭐가 있는지 찾아보는 것)

np.load 함수를 써서

loadtxt, savetxt: 직접 해볼 것(regression.csv)
csv: comma seperated values(모든 것이 콤마로 분리 됨)
skiprows=1: 처음 것 빼기
dtype: 처음은 X, 두 번째는 Y라고 하자고 적음
formats

in[34]
random data의 shape을 5 by 2 by 3로 만듦
length: 5
shape이랑 dimension이 중요
size: 총 element의 개수
dtype: default로 만들어진 것

같은지 작은지 비교하기 위해서는 두 개의 차원이 같아야함

in[38]
numpy 속에 들어있는 sum 함수를 씀- a를 sum하라
a 자체가 numpy의 산물이기 때문에 a.sum()이라고 써도 가능함(함수라서 반드시 괄호가 있어야 함, 자기 자신이기 때문에 괄호 안에 무언가를 적을 필요는 없음)

in[39]
첫 번째 차원의 값들을 더함(위에서 아래로)

191105

np.array 한 다음 리스트를 넣어주면 numpy 형태의 data로 바뀜
다차원의 array로 만들 수 있음-> 계산 가능

pure tone의 합이 복잡한 sound 만듦-> sinu soidal
phasor
radian
π: 180도
sin( ), cos( )에 들어가는 입력값은 degree가 아닌 radian
sin 0 = sin 2π
그래프만 외우기
0부터 100π까지 그래프를 그리면 총 몇 번의 반복? 50번 (2π가 반복되기 때문, sin이든 cos이든)
cos(2분의3π)= 0 sin(2분의3π) = -1

오일러공식
e: 상수값(2.71...) i: imaginary의 약자, 허수

세타값이 변하면 결과가 달라짐

모든 수를 포함하는 카테고리: 복소수(실수와 허수 모두 포함, a+bi)

벡터: 숫자열
a+bi에서 (a,b)도 벡터값

코사인 그래프는 1부터, 사인 그래프는 0부터 시작
a축: 코사인과 똑같음(1부터 시작해서 내려갔다 올라갔다)
b축(허수부분): 사인과 똑같음(0부터 시작해서 올라갔다 내려갔다)

sin(Theta)에 시간의 개념이 들어갈까(e.g. frequency(초당 몇 번 왔다갔다하는지))? 안 들어감(각도값이기 때문, 몇 초에 몇 바퀴를 도는지는 안 들어감)
소리라는 실체는 반드시 시간의 개념이 필요함, 단순한 sin(Theta)는 실체의 소리가 될 수 없음

amp: amplitude, 진폭
freq: frequency, 단위 Hz, 1초에 몇 번 왔다갔다 하는지
sr: sampling rate, 단위 Hz, 음질 상 얼마나 고해상도인지

in[161], [162]가 중요
e-04: 10의 4승, 10000 (1.000e-04=10000분의 1초, 4.998e-01=0.4998)

time을 먼저 만들어내고 세타값을 통해 time과 연동시켜 phase로 바꿔줌
2*np.pi=2π

plot 함수는 두 개의 입력값을 받음(x와 y값)
t: time　　s: sine의 결과값

exp: 오일러

191107

sound의 library
as: 줄임말
큰 library의 이름: matplotlib (첫 번째 줄과 똑같은 역할을 하는 것: import matplotlib.pyplot)

parameter setting: 변수를 미리 세팅해둠
각각의 변수를 바꾸면 다른 것을 고칠 필요 없이 쉽게 바뀜

0부터 2파이까지 만든다고 할 때 제일 1값은 0, 제일 마지막 값은 2*pi - 각도값이 정의됨 (radian이 단위)
s로 받은 7개의 결과값이 나옴-> plot하면 사인 곡선이 나옴

figure: function(괄호가 있기 때문)
add_subplot: figure에 담긴 함수
-> 전체 figure를 만들고, ax라는 변수로 받고 (figure는 전체 화면)
subplot안 숫자(221): 2 by 2로 나누는데(2행 2열), 그 중 첫 번째(첫 번째 네모)
y축: 사인함수의 결과(태극 문양처럼 이어짐, 0부터 시작)
line처럼 생겼을 때 equidistant- x축과 y축의 관계(캡쳐한 이미지는 non-linear, default는 line('.'을 없앰))
곡선이 나타난다는 말 자체가 x의 equidistant한 것이 y축에는 반영 안됨

sr*dur: duration이 비율로 반영, time tick을 먼저 만듦 (in[19])
실제로 plot할 때는 theta가 아니라 time이 x축에 들어감(실체의 소리에 필요한 정보)
점들의 개수: 1000개(시험에 나올 수 있음)(in[28])

np.exp, 1j는 고정이고 theta만 바뀜
c를 print했을 때 하나하나의 값이 엄청나게 긴 벡터, 전부 a+bi의 형태
-01: 10분의 1, -02: 10의 제곱분의 1
표기방법이 같음- 쓰는 숫자, 정보의 양이 똑같음 (컴퓨터는 정보량을 정해야함)

plot에 입력값이 3개 들어감- 3차원(한 점이 3차원 벡터가 됨)
1000개의 점이 찍힘
c.real: a만 빼오는 역할(e.g. 0.99802673)(복소수를 real part와 imaginary part로 분할하여 받아옴)

오일러 공식에서 코사인: real part와 관련, 사인: imaginary와 관련

ipd: 오디오를 재생하기위해 import(입력값: 시그널과 sr)

191112

사인, 코사인은 시간 없이도 만들어짐(세타값만 있어도 커브는 만들어짐), 그것만으론 실제 소리를 만들 수는 없음(소리는 시간의 개념이 반드시 들어가야 함)

sounddevice: 한번 깔아줘야 함

amp를 2로-> 더 커짐

sampling rate와 frequency가 연결되는 부분이 있음
sr이 100Hz일 때-> 우리가 표현할 수 있는 숫자는 1초에 100개
이 100개의 숫자로 1Hz의 frequency를 표현할 수 있나? ㅇㅇ있다(한번의 sine wave 주기가 있으면 됨)
2Hz는? 2번 왔다갔다

100개의 숫자로 10000Hz?(1초에 10000번, 주어진 숫자 100개) 안 됨, 숫자가 너무 적음-> sr이 충분히 있어야 그만큼의 주파수 표현 가능
우리가 표현할 수 있는 주파수는 주어진 숫자의 개수의 반까지 밖에 안 됨-> sr의 반까지 표현할 수 있음(sr이 10Hz면, 표현할 수 있는 frequency는 최대 5Hz)- nyquist frequency는 sr의 2분의 1
-> 우리가 표현하고자 하는 것보다 sr을 2배로, 넉넉하게

Fend: 제일 마지막(sampling rate의 반, nyquist frequency)
s는 계속 더해지고 업데이트 됨
제일 처음 s가 정의되지 않기 때문에 error가 뜸-> 처음 s가 뭔지 위에서 정의해줘야 함(s=np.zeros(len(t)))

pulse train: sine wave같은 부드러운 부분이 없어짐- sr이 많으면 선-0의 반복이 될 것-> pulse train이라 부름
그래프: nyquist까지 더했을 때의 그림

191114

Spectrum 역시 만들 수 있음
spectrum은 타임을 나열하는 것이 아닌 한 타임 안에 있는 주파수를 보여주는 것
Def 로 만들고 싶은 function을 정의할 수 있음
return은 출력을 의미
이후 기본 source에 vocal tract가 지나면 사람이 발음하는 소리처럼 나타남, 이는 RG와 BWG를 통해서 만들 수 있음

191119

기계: 인공지능(함수)
앞부분과 뒷부분의 데이터는 벡터(숫자열)로 되어 있어야 함: 벡터-> 기계(행렬의 형태)-> 벡터
앞부분 text-> 기계-> 뒷부분 음성으로 나옴: 음성 합성
일본어 text-> 기계-> 한국어 text: 기계 번역
중간의 기계는 행렬의 형태를 지님
인공지능: 행렬의 곱, 입력 벡터를 출력 벡터로 바꿔주는 함수의 역할(벡터는 음성, 텍스트, 이미지 등 가능)
선형대수: 행렬

linear algebra

행렬은 직사각형 형태로 생김
dimension: 차원, 행렬의 크기를 말할 때 사용됨
m행 n열(m by n 행렬)

벡터: 행렬의 일종
1 by n 행렬
벡터는 가로로 길게 있든 세로로 길게 있든 sequence of numbers
길게 생긴 벡터: column vector
차원이 늘어난다고 점이 늘어나진 않음

벡터 스페이스
linear combinations(중요!): v와 w는 차원이 같은 두 벡터, c와 d는 scalar
단순히 곱하고 더하면 linear combination

2차원의 벡터 스페이스: 모든 가능한 벡터가 다 덮여진 공간
1사분면만 덮어진 것은 하나의 완성된 스페이스라고 할 수 없음- 모든 것을 커버하지 못함
벡터 스페이스는 1차원, 2차원, 3차원의 모든 공간

n차원의 공간은 n개의 component(3차원의 공간은 3개의 component)가 공간을 다 채워야 함

column space: column vector로 모든 공간을 채울 수 있음
column space는 column vector보다 높은 차원일 수 없다(2차원)
원점과 두 칼럼 벡터(같은 선상에 있지 않은 independent한 벡터)를 연결하면 삼각형이 만들어짐- 평면 위에 있고 쭉 확장될 수 있음-> column space
whole space: 벡터 자체가 갖고있는 space, 2차원 그대로
column space도 그것을 다 채움-> 2차원
같은 선상에 있으면 dependent-> 아무리 곱하고 더해도 같은 선상에만 있어 whole space를 다 쓰지 못하고 column space는 1차원에 불과
whole space의 dimension은 몇 개의 row가 있는지 찾으면 됨
column space의 dimension은 몇 개의 independent한 column이 있는지 찾으면 됨

whole space는 3차원인데 column space는 2차원- P
직사각형도 얘기할 수 있음
column vector가 2개(3차원 속에서 찍힘)- whole space는 3차원, independent한 column은 2개라 column space는 2차원

transpose: 행렬을 뒤집음
2 by 3 행렬이 됨
whole space: column vector가 2차원
column vector가 2차원 상에서 3개가 있음- 벡터가 몇 개든지 간에 무조건 2차원
whole space는 달라지지만 column space는 여전히 2차원

m by n matrix가 있다면 whole space는 2개
(1,2,3)과 (2,4,6)의 column space는 1차원
whole space는 3차원인데 column space는 1차원, 나머지 빈 공간은 null space(의미 없는 공간)
null space: whole space에서 column space를 정의하고 난 나머지, 어떤 행렬이 있을 때 무엇을 곱하든지 간에 반드시 0이 되는 모든 가능성이 되는 공간
null space는 column space와 row space 모두에 존재할 수 있음

column vector로 접근했다면 얼마나 dependent하고 independent한지

행렬은 대문자로, 벡터는 소문자로 씀 (x가 입력 벡터, b는 출력 벡터, A라는 행렬과 곱해져서 출력됨)
입력 벡터와 출력 벡터의 차원은 항상 같을 필요는 없음, 달라질 수도 있음- 그 역할을 해주는 것이 행렬
A는 transformation matrix

함수에 넣어 선상으로 합쳐진 벡터를 역함수하면 점이 원래 자리로 돌아갈 수 없음(not invertible)
determinant는 row vector와 똑같다..?

eigenvector(중요!!)

191121

행렬 곱하기에서는 인접한 두 숫자가 같아야 곱해짐(e.g. x by 2, 2 by y, 둘을 곱하면 x by y의 형태가 됨)
transpose- 결과값도 transpose됨(계산해보면 나옴)

columnize- 세 점을 찍으면 plane(column space) 위의 삼각형이 됨, 이를 무한대로 밀면 결국 전체를 커버하게 됨-> spanning, 이 때 column space는 whole space와 다름(2차원 vs 3차원)
남은 1차원은 null space

spanning: linear combination으로 표현가능한 모든 것

rowize의 whole space: 숫자를 2개 쓰니까 2
row vector가 만드는 space는 2차원을 넘어가지 않음- whole space가 2차원이라 2차원보단 작거나 같아야 함

rank: independent한 column의 개수
row vector는 3개
independent한 것은 나머지의 linear combination으로 만들어질 수 없는 것이어야 함
column으로 보든, row로 보든, independent한 벡터의 숫자는 같음
column space의 개수=row space=independent
남는 것은 null space, null space는 column과 row 각각 다를 수 있음

whole space- 3차원(ppt)

linear combination: 어떤 두 벡터를 앞에 scaling 해서 더하는 것
subspace: whole space에 포함되어있다는 말
row space와 column space의 차원은 항상 같아야 함
independent한 벡터 2개-> rank 2, span된 column space도 2

P: plane, 2차원
L: line, 1차원

detransformation: 역행렬