데이터: 반드시 숫자화 되어야 한다.
데이터 종류: 소리, 영상, 텍스트, 숫자 데이터
벡터: 숫자의 열을 벡터라고 한다. (행렬과 달리 한 줄에 길게 쭉 나열해놓은 것.)
사진을 확대해서 보이는 하나의 픽셀은 각각 하나의 숫자다.
행렬: 숫자를 직사각형 안에 넣어놓은 것. 사진은 행렬의 형태다.
모든 데이터는 행렬으로 벡터로 전환할 수 있다. 사각형 안에 넣을지, 한 줄로 할지 선택하면 되고, 데이터는 일반적으로 벡터일 때 더 다루기 쉽다.

영상: 기본적으로 2차원. 여기에 컬러 이미지면 3차원, 여기에 시간까지 도입하면 4차원.
소리: 벡터화 할 수 있다. waveform의 소리를 한 줄로 나열하면 그것이 벡터화다.
텍스트: 50000개의 단어가 있다고 가정하면, 첫 단어를 10000...000, 두 번째 단어를 0100.....000, 마지막 단어를 0000...00001의 방식으로 벡터화할 수 있다.

***라이브러리**
파이썬에서 중요한 함수들을 정리해놓은 집합. 하나의 라이브러리 아래에 하위의 라이브러리가 존재한다.(import로 불러와야한다.)
형식: import numpy(필수!!!) as(선택)
import numpy from a를 통해서 하위의 함수로 바로 접근 가능 **시험!! 문법!**
ex. numpy안의, a안의, d 라이브러리의 함수: numpy.a.d.함수
***numpy란**
a = [1, 3, 5]
b = [2, 4, 6]
c = a+b
c [1, 3, 5, 2, 4, 6] 즉, 수학적 처리(사칙연산)가 되지 않고 단순 나열 형태로만 나옴
이때, 사칙연산이 가능하게 하는 것이 numpy임.
형식: import numpy (필수!!!)
A = numpy.array(a); B = numpy.array(b); A+B
import numpy as np (numpy를 np로 줄일 수 있음)
type(A) numpy.ndarray

X=np.array([[1,2,3],[4,5,6]])
x

11/19 ai와 데이터
음성의 데이터 형태들: 음성(벡터), text(벡터)
음성을 입력하여 text출력: 음성인식
입력, 출력되는 데이터는 모두 벡터 형태다. 그 중간과정에서 기계(인공지능, 또는 함수)는 행렬 형태로 데이터를 처리한다.
그렇다면, 그 벡터를 행렬로 처리하는 과정을 배우자.
인공지능 또는 기계 또는 함수: 행렬의 곱셈과정을 통해, 입력 벡터를 출력 벡터로 바꾸어주는 함수 역할. ---> “선형대수”

선형대수: Linear Algebra
행렬: Matrices: 행렬의 크기 = 행렬의 차원 = dimension
vectors: 1행의 행렬 or 1열의 행렬(한 줄) = a sequence of numbers
column matrices(열 행렬): 1열의 행렬(x,y 좌표 상에 점으로 표현 가능)
“행렬의 곱을 수치로 계산할 수 있는데, 동시에 그림에도 표현 가능.”(vector multiplication, addition 참고)

*******linear combination******
: 곱셈과 덧셈으로 이루어진 식(제곱 등등 없이)-> cv+dw의 형태
vector space: 여러 벡터들이 만들어낸 공간. 무한한 가능한 벡터(점)들로 채워진 공간
R: 1차원 공간 R의 제곱: 2차원(2행 1열) R의 n제곱: n차원 공간.(n행 1열 데이터 기반)
A = 2 -1 2 -1
    1 3을 1 3으로 나눠 무한의 linear combinations을 하면, 결국, 2차원 공간을 만들 수 있다. 즉, 3차원까지 확장되진 않는다.
slide29: col1과 col2는 다른 선상에 있으므로 서로 independent 하다고 한다.(원점과 col1,2는 삼각형 이룸.) col1, 2는 linear combination을 통해 각각의 선을 무한히 채울 수 있다.
slide37의 col1, 2: dependent (동일 선상).-> whole space: R의 제곱 Column space: L(1차원)
즉 column space가 independent하면 2차원, dependent하면 1차원이라고 볼 수 있다.

slide40의 3x3행렬: column space: R세제곱
41: col1,2가 서로 곱셈 관계이므로 P(plain, 2차원)
**42: 1*col1+ 1*col2 = col3인 경우.-> P 43: L(line, 1차원)
44: 열 행렬이 3개라서 전체 공간은 3차원, 하지만 점이 두 개 이므로 2차원을 못 벗어남.
44->45: Transpose: 3*2행렬의 데이터를 그대로 2*3행렬로 옮기는 것.
44와 45의 차이: whole space, column space는 항상 2차원이다!!!
m*n행렬은 두 개의 whole space: “R의 m제곱, R의 n제곱”을 가진다.(row/column space)
49: column space는 3차원, row space는 1차원(dependent)이므로, 그 사이의 공간이 남는데, 그 나머지 빈 공간을 통칭하여 “Null Space”라고 부른다.
50: Null space의 수학적 정의. 이때의 x1 x2를 null space
55: 총 정리

****Linear transformation***
Ax = b -> x가 입력벡터, b가 출력벡터. 입력벡터와 출력벡터의 차원이 꼭 같을 필요 x
ex) 1*4 행렬 곱하기 4*3 행렬: 에서 출력 벡터는 3차원(뒤 행렬의 열 수)
A: Transformation Matrices
62-65: 61의 계산식을 기하학적으로 해석한 것. -> transformation matrices의 역할
78 inverse matrix: 역함수를 취한 것****example 3: dependent한 경우!!! 주목!!
역함수를 취할 경우, independent할 경우, transformation matrix로 돌아갈 수 있으나, dependent할 경우 특정할 수 없다. 즉 invertible하지 않다!

*11/21
3*2행렬에 곱할 수 있는 행렬: 2*x. 즉 앞 열과 뒷 행이 같아야 함. 결과: 3*x 행렬.
Transpose도 해보고,
whole space/null space/column space: dimension 구분!
spanning: 두 independent column vector를 원점과 연결하여 column space를 만드는 것 (기하학적 해석) = linear combination
null space: column space와 수직인 것. = whole space - column space
spanning의 한계: spanning은 절대 whole space의 dimension을 넘어갈 수 없다.
*(1,2) (-1,0) (3,5) 세 개는 상호 independent할까? -> 아니다.
-> a*x1 + bx2 = x3로 표현 가능하면 independent 아님.
->기하학적 해석: whole space - column space = 0이므로, null space가 없다.
하나의 벡터는 몇 차원이든지에 상관없이 무조건 하나의 점이다.
requirements to become a vector space: linear combinations still stay in the space
R의 n제곱 space consists of all vectors with n components.
columns of A의 m*n제곱 in R의 m제곱 all their linear combinations form a subspace: column space, C(A)
column space dimension=row space dimension -> rank도 동일=independent columns
dim(whole space) n rows dim(column space) n of independent columns
***시험: 41의 column space가 plain인 이유: (1,2,4)*2=(2,4,8) (slide 41~45)**
orthogonal
50: (right) null space 52: left null space

58: Ax = b 행렬: 대문자 벡터: 소문자 x: 입력벡터 b: 출력벡터. x의 차원과 b의 차원은 반드시 같을 필요는 없고, 차원은 A에 의해 정의된다. (59 ~ 77: linear transformation 연습)
78 ~ 97: detransformation, 역함수, 역행렬 연습. **97이 중요! (cf) 행렬의 determinant)**

*eigenvector: 어떤 행렬의 eigenvector은 무엇인가?: 어떤 행렬의 eigenvector은 transformation의 결과값이 원점과 같은 선상에 놓인다.