《 English Phonetics 》

2016131006 불어불문학과 이수연

● Phonetics : Speech(사람이 하는 말에 한정)을 연구하는 학문

- Articulatory phonetics(from mouth : producing) , Acoustic phonetics(through air : transmitting) , Auditory phonetics (to ear : hearing)

● Articulation

5 speech organs = constrictors = articulators

⑴ Phonation process : larynx (voicebox) - vocal cords vibrates

— voiced (유성음) : can feel vibration , 모든 모음과 일부 유성 자음 ex) v, z, l, m, a, i, …

— voiceless (무성음) : cannot feel vibration ex) f, s, k, p, h, …

⑵ Oro-nasal process : velum(soft palate) -

— Velum lowered = nasal sounds : [m], [n], [ŋ] , 숨 쉴 때

⑶ Articulatory process : lips, tongue tip, tongue body

● Control of constrictors(articulators)

⑴ Constriction location (CL) : where exactly? 앞뒤

— lips → bilabial, labiodental

— tongue body → palatal, velar

— tongue tips → dental, alveolar, palato-alveolar

⑵ Constriction degree (CD) : how much exactly? 상하

— upper part > … > lower part :

stops [p] , [b] , [t] , [d] , [k] , [g]

> fricatives [f] , [v] , [s] , [z] , [θ] , [ð] , [ʃ] , [ʒ] , [h]

> approximants [w] , [r] , [l] , [j]

> vowels

● Phonemes : Individual sounds that form words

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 → m, n, ŋ

larynx → p, f, θ, t, s, ʃ, k, h

● Acoustics

— intensity (dB) : pitch와 독립적으로 O

— pitch (Hz) : 성대가 1초에 몇분 떨렸는지

· male : 65-200Hz / female : 145-275Hz

— formant (Hz) : 모음을 구별하는 수치적인 지표

cf. pitch 와 formant의 차이

Pitch is the fundamental frequency of vibration of the vocal folds, which are present at the top of one's trachea. They vibrate quasi-periodically only for voiced phonemes, namely vowel, semivowel and nasal sounds. So, for unvoiced stops such as /p/, /k/, /t/, /th/, /ch/ and unvoiced fricatives such as /f/, /s/, etc. there is nothing called pitch.

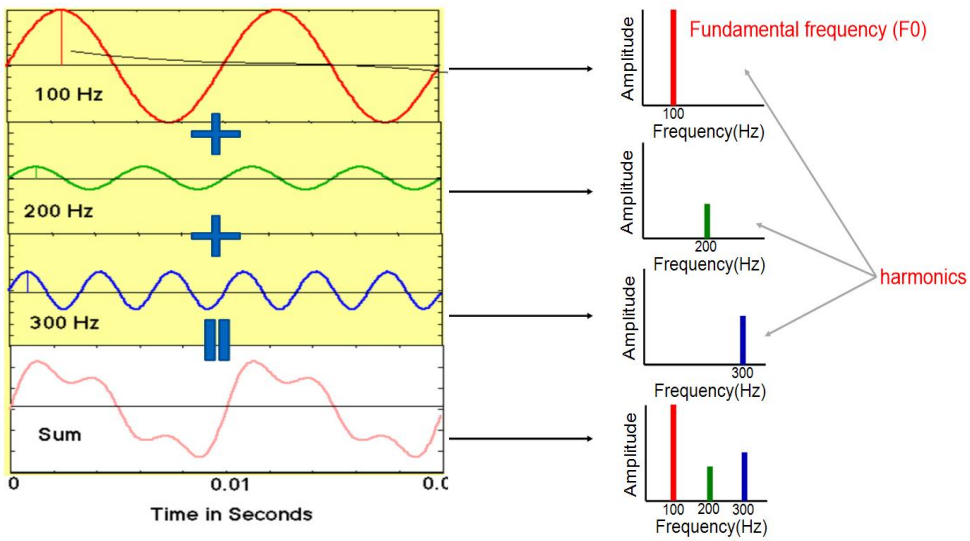
The formant frequencies are due to the frequency shaping of the signal from the vocal folds by the vocal tract. Vocal tract is everything from nasal tract, tongue, teeth, lips, palate, etc. The particular configuration of the above organs (articulators) for every phoneme creates resonances at specific frequencies called formants. So, formants exist for both voiced and unvoiced sounds.

Pitch can be estimated by quantifying the period (using autocorrelation, say) or measuring the harmonics. Formant frequecies can be found by linear prediction analysis from the poles.

★★★ 모든 (복잡한) 사운드는 여러개의 (단순한) sign wave의 결합으로 표현된다

complex tone simplex tone

ex)



↳ simplex tone을 더 단순하게 표현하는 그래프

★★★ x축 : frequency + y축 : amplitude ⇒ 스펙트럼 (사운드)

● Human voice = 여러 가지 sign wave가 합쳐져서 만들어졌지만 패턴이 있음.

sign wave 배수로 더하기더하기...

f0가 정해지고 그 대응의 합으로 voice source가 정해진다

pitch는 첫번째 프리퀀시와 일치 = f0

단위는 Hz.

EGG = 성대에서 직접 녹음한 소리

이건 스펙트럼 그림 점점 디크리즈 형태

그냥 오디오는 산맥 모양

'아'라고 할때 누가하든지 산맥의 패턴은 똑같음

첫 산맥 = 첫 포먼트

반복주기 = 우리가 만들었던 열개의 사인웨이브중에서 첫번째와 일치. 음높이도 100Hz와 일치. 1000hz는 인지적으로 들리지 않음. 합치면 합칠수록 부드러웠던 게 점점 뾰쪽뾰쪽 . 한쪽 피크가 높음. 무한대로 가면 피크하나 영영영... 같은모습으로 됨. 하나 나타난걸 펄스라고 부름. 펄스트레인.

F1 은 그 모음의 높낮이를 결정함. height

F2는 front back 을 결정. backness / frontness

10/01 기초 파이썬

코딩 = 자동화

모든 랭귀지는 단어가 있음. combine을 어떻게 하느냐에 따라 정보가 됨. > '단어' + 'combine' => communication

컴퓨터 랭귀지 = 사람 랭귀지

단어의 특징 : 의미(정보)를 담는 그릇.

컴퓨터 랭귀지의 단어 = "변수" variable

컴퓨터 랭귀지의 공통적 문법 = 1. 변수(그릇)에 정보를 넣는 것(assign) , 2. conditioning 조건 (if문) , 3. 반복loop (for문) , 4. 함수 (가장 중요) : 1~3을 패키징

= : 오른쪽 정보를 왼쪽 변수에 assign (equal 아님!)

a = 1이 아니라 1을 a라는 변수에 넣는다 라는 의미 (a는 1이라는 정보가 있구나)

파이썬에서 모든 함수는 누군가 만들어놓아야함 (직접 만들 수도 있음) . 아나콘다는 유용한 함수를

모아놓은 패키지(아마도 라이브러리?)

print() : 괄호 안에 variable 입력

정보의 종류 : 숫자, 문자

한 칸 = cell

헤더부분 선택하고 a 누르면 위에 셀 생성, b 누르면 아래 생성, x 누르면 삭제

variable 이름은 unique. 새로운 정보 입력하면 overwrite

실행 단축키= 쉬프트 + 엔터

b = love 와 b = 'love'의 차이 : 전자는 love라는 variable에 정보를 지정해야 오류가 안 남. 후자는 문자열

a = 1

b = 2

b

c = 3

c

이라고 입력하면

3 출력

: 가장 마지막에 변수명 하나만 치면 print out

엔터 안 칠 때는 ;

a = 1 ; b =2 ; c=3

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

정보를 한꺼번에 모아서 넣을 때 '리스트'

a = []

type() : 타입 확인

int = 정수(integer 철자이거맞나...)

float = 실수

str = 문자열(string)

리스트는 반드시 숫자만 들어갈 필요 노노 정수든 실수든 문자열이든.. 리스트 속에 리스트도 들어갈 수 있음

a=[1, 'love', [1, 'bye']] 이거도 리스트! items : 3개 (int, str, list)

a = () 이건 튜플. 리스트랑 똑같. 튜플은 더 보안에 강하다. 바꾸기가 힘들다.

a = {'a':'apple'} 딕셔너리. 표제어와 설명, 늘 쌍으로 되어있음(:)

type(a) = dict

10.08 String

In [25]:

```
a = {"1":"apple", "b":"orange","c":2014}
print(type(a))
print(a["1"])
```

```
<class 'dict'>
apple
```

In [11]:

```
a = 1 ; b = 1 ; c=a+b
c
```

Out[11]:

2

In [14]:

```
a = [1,2] ; b=[3,4] ; c=a[1]+b[1]
c
```

Out[14]:

6

In [24]:

```
a='123,124'
print(a[0])
```

1

In [27]:

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

Out[27]:

tuple

In [31]:

```
s='abcdef'
n=[100,200,300]
print(s[0],s[5],s[-1],s[-6])
print(s[1:3],s[1:],s[:3],s[:])
len(n)
```

```
a f f a
bc bcdef abc abcdef
```

Out[31]:

```
3
```

In [37]:

```
s = ' this is a house built this year.₩n'
s.rindex('this')
```

Out[37]:

```
23
```

In [39]:

```
tokens = s.split(' ')
s = ','.join(tokens)
s
```

Out[39]:

```
',this,is,a,house,built,this,year.₩n'
```

10.10 Syntax

In [1]:

```
# For Loop
a = [1,2,3,4]
for i in a: # in 뒤에 있는 것을 i로 하나씩 받아서 for 이하 시행
    print(i)
```

```
1
2
3
4
```

In [2]:

```
a = [1,2,3,4]
for i in range(4): #range(4) = "4개의 인덱스를 만들어줘"
    print(a[i])
```

```
1
2
3
4
```

In [3]:

```
a = [1,2,3,4,5,6,7]
for i in range(len(a)):
    print(a[i])
```

```
1
2
3
4
5
6
7
```

In [7]:

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

```
red
green
blue
purple
```

In [11]:

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

for i, s in enumerate(a): # 함수 enumerate(a) : a의 리스트에 번호를 추가로 매김. 0부터. 첫번째 변수
    print(a[i])
```

```
red
green
blue
purple
```

In [13]:

```
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)) # "   " 안의 형태로 적고 싶을 때, varialbe 개수만큼 {} 쓰면
```

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

In [14]:

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

for i,s in zip(a,b): #zip : 두 리스트를 하나로 합침. 길이가 같아야함(?). 두 리스트가 페어로 존재하
     print("{}: {}%".format(i, s*100))
```

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

In [25]:

```
a = 1
if a == 0 : # a가 0이면
    print("yay!")
if a != 0 : # a가 0이 아니면
    print("non!")
if a >= 0 : # a가 0보다 크거나 같으면

else: #"아니면"
    print("no")
```

```
  File "<ipython-input-25-21efdbdf2719>", line 8
    else: # "아니면"
        ^
IndentationError: expected an indented block
```

In [27]:

```
for i in range(1,3):
    for j in range(3,5):
        print(i*j)
# 이런건 무조건 시험... 두 번 for loop 도는 거!
# 가장 바깥쪽 루프(i)는 2번 돌고 j에서 또 2번씩 도니까 총 4번 실행
```

```
3
4
6
8
```

In [28]:

```
for i in range(1,3):
    print(i)
    for j in range(3,5):
        print(i*j)

#첫번째 프린트 2번, 두번째 프린트 4번
```

```
1
3
4
2
6
8
```

In [29]:

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

```
4
8
```

In [34]:

```
for i in range(1,3):
    if i >=3:
        for j in range(3,5): #if 아래는 항상 indent. 안되면 오류!
            print(i*j)
```

10. 29 Numpy

In [9]:
```
a = [1, 3, 5]
b = [2, 4, 6]
a+b
```

Out[9]:
```
[1, 3, 5, 2, 4, 6]
```

In [11]:
```
import numpy as np
```

In [7]:
```
A = numpy.array(a)
B = numpy.array(b)
```

In [8]:
```
A+B
```

Out[8]:
```
array([ 3,  7, 11])
```

In [10]:
```
type(A)
```

Out[10]:
```
numpy.ndarray
```

In [13]:
```
X = np.array([[1,2,3],[4,5,6]])
```

In [14]:
```
X
```

Out[14]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [15]:
```
X.shape #차원 의미. 2 x 3
```

Out[15]:
```
(2, 3)
```

10.31 Numpy

In [3]:
```
import numpy as np
#from numpy import A # numpy 안에 있는 A를 가져오자
import matplotlib.pyplot as plt
#A.B = A 패키지에 있는 B
#시험문제에 바로 나올 수 있는 것
#from matplotlib import pyplot as plt 라고도 쓸 수 있음
```

In [7]:
```
np.empty([2,3], dtype='int')
```

Out[7]:
```
array([[         0, 1072168960,          0],
       [1072168960,          0,          0]])
```

In [9]:
```
np.zeros([2,3])
```

Out[9]:
```
array([[0., 0., 0.],
       [0., 0., 0.]])
```

In [12]:
```
np.array([[0,0,0],[0,0,0]])
```

Out[12]:
```
array([[0, 0, 0],
       [0, 0, 0]])
```

In [15]:
```
np.ones([2,3])
```

Out[15]:
```
array([[1., 1., 1.],
       [1., 1., 1.]])
```

In [18]:
```
np.ones([2,3], dtype='int')
```

Out[18]:
```
array([[1, 1, 1],
       [1, 1, 1]])
```

In [19]:
```
np.arange(5) #range()랑 비슷
```

Out[19]:
```
array([0, 1, 2, 3, 4])
```

In [20]:
```
np.arange(0,10,2, dtype='float64')
```

Out[20]:
```
array([0., 2., 4., 6., 8.])
```

In [21]:
```
np.linspace(0,10,6) #linear space. 처음과 끝을 포함하여 6개로 똑같이 나누어준다 (10까지 포함인 모습)
# np.linspace(시작, 종료, 개수) : 개수에 맞게끔 시작과 종료 사이에 균등하게 분배
```

Out[21]:
```
array([ 0.,  2.,  4.,  6.,  8., 10.])
```

In [24]:
```
X = np.array([[1,2,3],[4,5,6]]) #[] 대괄호 두개 = 이차원 ; 대괄호 3개 = 3차원
X
```

Out[24]:
```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [33]:
```
Y = np.array([[[1,2,3],[4,5,6]],[[1,2,3],[4,5,6]]]) #2차원 두 개는 삼차원
Y
```

Out[33]:
```
array([[[1, 2, 3],
        [4, 5, 6]],

       [[1, 2, 3],
        [4, 5, 6]]])
```

In [28]:
```
X.ndim #차원 알려주는
```

Out[28]:
```
2
```

In [29]:
```
X.shape #2x3인 차원이다(?)
```

Out[29]:
```
(2, 3)
```

In [34]:
```
Y.shape #2x3이 2개인 차원
```

Out[34]:
```
(2, 2, 3)
```

In [31]:
```
X.dtype
```

Out[31]:
```
dtype('int32')
```

In [35]:
```
X.astype(float) #타입 변환
```

Out[35]:
```
array([[1., 2., 3.],
       [4., 5., 6.]])
```

In [36]:
```
np.zeros_like(X) # X*0해도 똑같음
# like(배열) 지정한 배열과 동일한 shape의 행열을 만듦
# 종류 : np.zeros_like(), np.ones.like(), np.full_like(), np.empty_like()
```

Out[36]:
```
array([[0, 0, 0],
       [0, 0, 0]])
```

In [38]:
```
data = np.random.normal(0,1,100) # 정규분포normal distribution로 데이터 랜덤생성 .
# np.random.normal(정규분포 평균, 표준편차, (행, 열) or 개수) : 정규 분포 확률 밀도에서 표본 추출
print(data)
```

```
[ 0.17220936  3.63413319  1.79004811  0.43605779 -1.33729651  0.62791974
 -0.27576677  1.59299385  0.6866562  -0.39559477  0.14257864 -0.94367155
  0.61241866  0.5192135   0.34373035 -2.51310141  0.27534836 -0.92343738
  0.41763106 -0.58722763  1.44240769  2.1774158  -0.51703999  0.30987555
 -1.15293726 -0.05220753 -0.95385663  1.1801642   0.9309844  -0.25005757
 -1.07752434 -0.65908623 -0.24064692 -0.13865326 -0.24709929  0.02401597
  1.27916544  0.10482293  0.74158559  0.18170936 -1.03909671 -0.42212284
 -0.44528969  0.29721601  0.08980602 -0.52085625  0.92360159 -0.48917233
 -1.02450491  2.28863031  0.68496109  0.64009845 -1.41452731  1.26460298
  0.4795454   0.89346115  1.02752001 -1.92346047  1.05196862  0.33130818
  0.20861423  0.29873275  0.80336811  0.29719907 -1.12231016 -1.92610588
 -1.34559993 -0.10650772  0.12018836 -0.13937052 -0.29864508  0.63569818
  0.82943184  0.73361811 -0.03781109 -0.30225504 -0.86248564  0.09599242
 -0.56847991  1.39996332  1.24408033 -0.33508835  0.85697196 -0.61379087
  1.28343712  0.27461376 -0.42293543 -0.37213719 -0.11331183  1.90095436
 -0.35644987  0.58728444  0.38027114  0.1546923  -0.57848473 -1.60583919
 -1.15062939  1.9892623  -1.09017288  0.25643626]
```

In [39]:
```
data.ndim
```

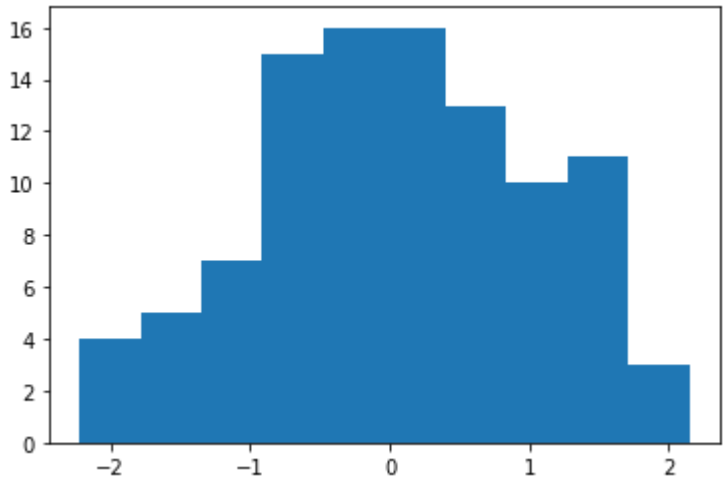
Out[39]:
```
1
```

In [40]:
```
data.shape #그냥 벡터니까. 백개의 데이터가 나왔다는 뜻
```

Out[40]:
```
(100,)
```

In [41]:
```
data = np.random.normal(0,1,100)
print(data)
plt.hist(data,bins=10) #bins 바구니. 그래프의 막대 몇 개인지 정함.
plt.show()
```

```
[-0.3919344   0.85328007  1.31792715 -1.14214277  0.45993339  1.19237434
 -1.02181608 -0.49631218  1.45856153 -0.30358796 -0.93243261  2.14510453
 -2.14206908  0.50933277 -1.32785307 -0.19605045 -1.32569246 -1.42286844
  0.12187997 -0.39704067  0.89102163  0.44603138 -0.44873535  0.62541739
  0.63974312 -0.43001933  1.2323329  -1.16539204  1.28185058  0.81240661
 -1.7697323  -0.05408883  1.394039   -0.34560812  1.40819354 -0.81996811
 -0.65906678  1.5137701   1.88070512  0.58407675 -0.45690393  0.86449837
 -0.31735365 -0.83041907 -2.22251954 -1.51265914 -0.18107539  0.8603055
 -0.76774571 -0.6719022   0.53482455  1.30357288  0.02749976  0.09892231
  0.94722325 -0.090717    0.10708013  0.05055964  0.78872663 -0.62910701
 -1.73355073  0.27333047  0.11956334  0.14437983  1.64404964 -1.04701272
  1.27026342 -0.03661248  0.01094321 -0.22536437 -1.58808479  0.7175769
  0.30770397 -0.53493031  0.92781105 -0.16238469  0.17518897  0.54869091
  0.0344384  -1.9409634  -2.11601669  1.70699606 -0.6335011  -0.46288451
  1.80403256  0.32196506  0.61978296  1.53970266 -0.06353558 -0.80258417
 -0.78887344  0.26148957 -0.71720822 -0.56103709  0.85281175  0.82902079
 -0.69590287  0.37296518  1.38004226 -0.6182597 ]
```



2. Manipulation

In [43]:
```
X = np.ones([2,3,4]) # 2x3x4개 데이터
X
```

Out[43]:
```
array([[[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]],

       [[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]]])
```

In [44]:
```
Y = X.reshape(-1,3,2) #같은 차원에서만 변경 가능. 니가 알아서 해라라고 할 때 첫번째에 -1이라고 함.(4,3,2)랑 같은결과
Y
```

Out[44]:
```
array([[[1., 1.],
        [1., 1.],
        [1., 1.]],

       [[1., 1.],
        [1., 1.],
        [1., 1.]],

       [[1., 1.],
        [1., 1.],
        [1., 1.]],

       [[1., 1.],
        [1., 1.],
        [1., 1.]]])
```

In [ ]:
```
np.allclose(X.reshape(-1, 3, 2), Y)
#어레이해서 두 개 비교 . assert는 몰라도 됨
```

In [49]:
```
a = np.random.randint(0,10,[2,3])
b = np.random.random([2,3])
np.savez('test',a,b) # savez() 실제 파일로 저장해줌
```

In [53]:
```
del a,b  # print all interactive variables 메모리 전체 삭제
%who # show available variables now
```

```
X       Y       data    np      numpy   plt
```

In [57]:
```
npzfiles = np.load("test.npz") # 불러오기
npzfiles.files
```

Out[57]:
```
['arr_0', 'arr_1']
```

In [58]:
```
npzfiles['arr_0'] # 57행 결과. a 값. 'arr_1' 넣으면 b 값
```

Out[58]:
```
array([[6, 6, 8],
       [3, 5, 7]])
```

In [4]:
```
data = np.loadtxt("regression.csv", delimiter=",", skiprows=1, dtype={'names':("X", "Y"), 'formats':('f', 'f')})
#파일 불러오기 : np.loadtxt("파일경로",파일에서 사용한 구분자, 데이터타입 지정), data 변수에 array로 넣어준다
data
```

Out[4]:
```
array([( 3.3  , 1.7  ), ( 4.4  , 2.76 ), ( 5.5  , 2.09 ), ( 6.71 , 3.19 ),
       ( 6.93 , 1.694), ( 4.168, 1.573), ( 9.779, 3.366), ( 6.182, 2.596),
       ( 7.59 , 2.53 ), ( 2.167, 1.221), ( 7.042, 2.827), (10.791, 3.465),
       ( 5.313, 1.65 ), ( 7.997, 2.904), ( 5.654, 2.42 ), ( 9.27 , 2.94 ),
       ( 3.1  , 1.3  )], dtype=[('X', '<f4'), ('Y', '<f4')])
```

4.Inspecting

In [59]:
```
arr = np.random.random([5,2,3])
```

In [62]:
```
print(type(arr))
print(len(arr))
print(arr.shape)
print(arr.ndim)
print(arr.size) # 총 elements 개수
print(arr.dtype)
```

```
<class 'numpy.ndarray'>
5
(5, 2, 3)
3
30
float64
```

5.1 Arithmatic

In [66]:
```
a = np.arange(1,5)
b = np.arange(9,5,-1)
print(a)
print(b)
```

```
[1 2 3 4]
[9 8 7 6]
```

In [65]:
```
print(a-b)
print(a*b)
```

```
[-8 -6 -4 -2]
[ 9 16 21 24]
```

5.2 Comparison

In [67]:
```
a = np.arange(1,10).reshape(3,3)
b = np.arange(9,0,-1).reshape(3,3)
print(a)
print(b)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
[[9 8 7]
 [6 5 4]
 [3 2 1]]
```

In [68]:
```
a == b
```

Out[68]:
```
array([[False, False, False],
       [False,  True, False],
       [False, False, False]])
```

In [69]:
```
a > b
```

Out[69]:
```
array([[False, False, False],
       [False, False,  True],
       [ True,  True,  True]])
```

In [70]:
```
a.sum(), np.sum(a)
```

Out[70]:
```
(45, 45)
```

In [72]:
```
a.sum(axis=0) , np.sum(a,axis=0) # axis = n : n+1번째 차원에서 sum할 것인지. 여기서는 1차원 합이니까 같은 열끼리 합
```

Out[72]:
```
(array([12, 15, 18]), array([12, 15, 18]))
```

In [73]:
```
a.sum(axis=1) , np.sum(a,axis=1) # 같은 행끼리 합
```

Out[73]:
```
(array([ 6, 15, 24]), array([ 6, 15, 24]))
```

Broadcasting

In [75]:
```
a = np.arange(1,25).reshape(4,6)
a
```

Out[75]:
```
array([[ 1,  2,  3,  4,  5,  6],
       [ 7,  8,  9, 10, 11, 12],
       [13, 14, 15, 16, 17, 18],
       [19, 20, 21, 22, 23, 24]])
```

In [76]:
```
a + 100
```

Out[76]:
```
array([[101, 102, 103, 104, 105, 106],
       [107, 108, 109, 110, 111, 112],
       [113, 114, 115, 116, 117, 118],
       [119, 120, 121, 122, 123, 124]])
```

In [77]:
```
b = np.arange(6)
b
```

Out[77]:
```
array([0, 1, 2, 3, 4, 5])
```

In [78]:
```
a + b #행마다 b 더하기
```

Out[78]:
```
array([[ 1,  3,  5,  7,  9, 11],
       [ 7,  9, 11, 13, 15, 17],
       [13, 15, 17, 19, 21, 23],
       [19, 21, 23, 25, 27, 29]])
```