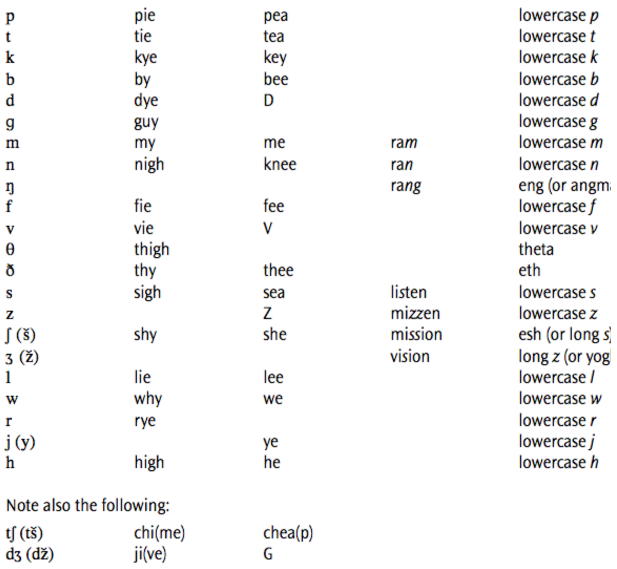
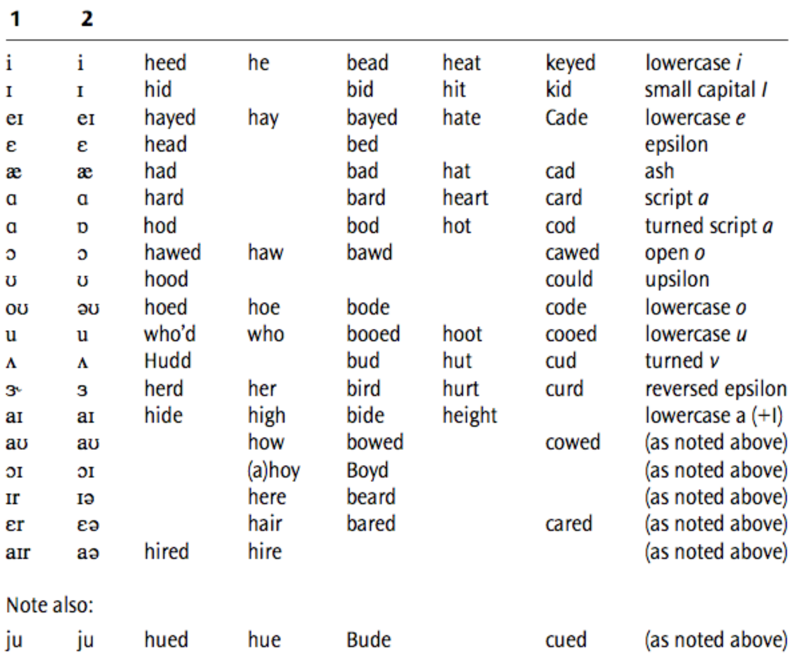
-English consonants and vowels



p	pie	pea		lowercase *p*
t	tie	tea		lowercase *t*
k	kye	key		lowercase *k*
b	by	bee		lowercase *b*
d	dye	D		lowercase *d*
g	guy			lowercase *g*
m	my	me	ra*m*	lowercase *m*
n	nigh	knee	ra*n*	lowercase *n*
ŋ			ra*ng*	eng (or angm
f	fie	fee		lowercase *f*
v	vie	V		lowercase *v*
θ	thigh			theta
ð	thy	thee		eth
s	sigh	sea	listen	lowercase *s*
z		Z	mizzen	lowercase *z*
ʃ (š)	shy	she	mission	esh (or long *s*
ʒ (ž)			vision	long *z* (or yog
l	lie	lee		lowercase *l*
w	why	we		lowercase *w*
r	rye			lowercase *r*
j (y)		ye		lowercase *j*
h	high	he		lowercase *h*

Note also the following:

tʃ (tš)	chi(me)	chea(p)
dʒ (dž)	ji(ve)	G



1	2						
i	i	heed	he	bead	heat	keyed	lowercase *i*
ɪ	ɪ	hid		bid	hit	kid	small capital *I*
eɪ	eɪ	hayed	hay	bayed	hate	Cade	lowercase *e*
ɛ	ɛ	head		bed			epsilon
æ	æ	had		bad	hat	cad	ash
ɑ	ɑ	hard		bard	heart	card	script *a*
ɑ	ɒ	hod		bod	hot	cod	turned script *a*
ɔ	ɔ	hawed	haw	bawd		cawed	open *o*
ʊ	ʊ	hood				could	upsilon
oʊ	əʊ	hoed	hoe	bode		code	lowercase *o*
u	u	who'd	who	booed	hoot	cooed	lowercase *u*
ʌ	ʌ	Hudd		bud	hut	cud	turned *v*
ɝ	ɜ	herd	her	bird	hurt	curd	reversed epsilon
aɪ	aɪ	hide	high	bide	height		lowercase a (+I)
aʊ	aʊ		how	bowed		cowed	(as noted above)
ɔɪ	ɔɪ		(a)hoy	Boyd			(as noted above)
ɪr	ɪə		here	beard			(as noted above)
ɛr	ɛə		hair	bared		cared	(as noted above)
aɪr	aə	hired	hire				(as noted above)

Note also:

ju	ju	hued	hue	Bude		cued	(as noted above)

-Phonetics: a study on speech

1. Articulatory phonetics(from mouth): how to produce speech
2. Acoustic phonetics(through air): how to transmit speech
3. Auditory phonetics(to ear):how to hear speech

-upper vocal tract: Lip, teeth, alveolar ridge, hard palate, soft palate(velum), uvular, pharynx wall, larynx

-lower vocal tract: lip, tongue tip, blade, front, center, back, root, epiglottis(음식이 기도로 가는것을 막음)

-larynx

1. Voiced: vibration- 성대가 붙어있을 때
2. Voiceless:no vibration-성대가 떨어져 있을 때

-velum lowered: nasal, breathing

-constriction

1.Constriction location

1) Lips-bilabial, labiodental

2) tongue body- palatal, velar cf)모든 모음은 tongue body를 사용

3) tongue tip- (inter)dental, alveolar, retroflex, palato-alveolar

2.Constriction degree

Stops>fricative>approximants>vowel

-phonemes: individual sounds that form words

* Spectrum

모든 소리는 spectrum으로 이루어져 있음.

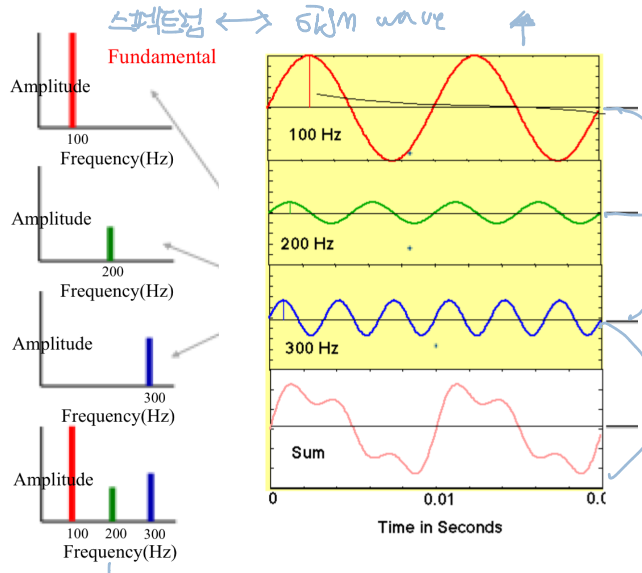
스펙트럼 ⟷ sign wave

time × value (frequency)

cf) Praat 이전용

spectrum : 숫자 입력,

또는거 : sign wave



simplex (Pure) tone

complex tone

: simplex tone을 합성한 결과.

frequency × amplitude (magnitude)

(주파수 × 그 주파수의 크기)

* Source (목구멍에서만 나는소리)

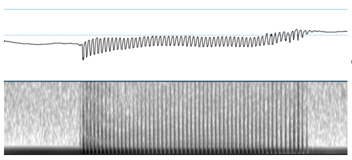
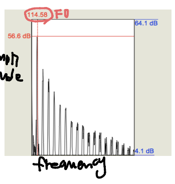
- Human voice source consists of *harmonics* 배수로 증가
- A complex tone = sum of pure tones at integer multiples of the lowest pure tone
- the lowest pure tone
  - *Fundamental frequency (F0)* = Pitch
  - rate of vibration of the larynx
  - the number of opening-closing cycles of the larynx per second (1초동안의 진동수)
- Amplitude of pure tones gradually decreases

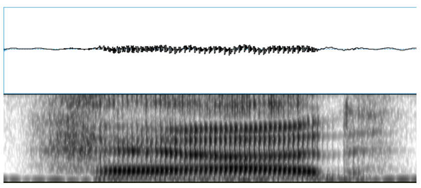
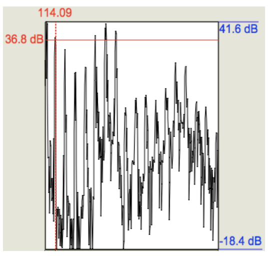
cf) 여 f(0) ↑ → 듬성듬성
남 f(0) ↓ → 빽빽.

* Filter

- Compare spectrums between audio and EGG
- EGG: gradual decreasing
- audio: peaks/mountains and valleys harmonics인건 그대로이나, amplitude가 gradual decrease가 아님. → carved.
- Because it is filtered by the vocal tract (VT)
  - peaks/mountains: frequencies VT likes = formants
  - valleys: frequencies VT does NOT like

source를 재는 기구.

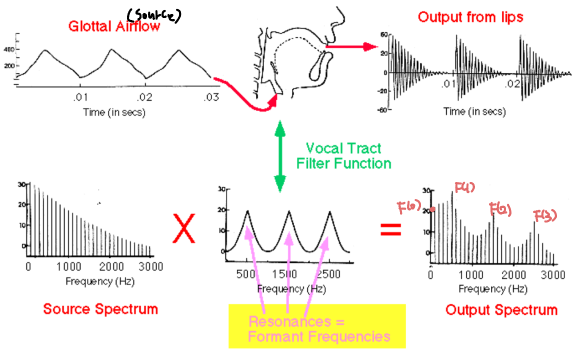



* Synthesizing Source

- convert to mono → 첫번째 sign wave와 주기일치, 높이도 첫번째 거다 비슷.
- 무한대로 가면, pulse train.

* Source-filter theory



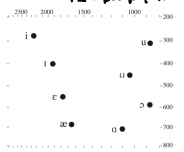
Formant: Peak의 frequency

* Spectrogram

- Dark band : mountains = Formants.

* F1, F2 만으로 모음구분 可

F2 (back? front?)



F1 (hight)

코딩 = 자동化 (ex. 쓸어있는 프로그램들)

언어 ~~[illegible]~~

사람의언어 [단어 : 정보 (단어는 정보를 담는 그릇)
≒ 문법

컴퓨터언어 [변수 : 정보 ex) 숫자, 문자...
문법
① 변수에 정보를 assign
② 조건 (if)
③ 반복 (for)
☆④ 함수 : 입력 → 출력

$x = y$ ← y를 x로 assign. ex) a=1 / $\boxed{\text{Run}}$ (단축키 : Shift + enter)

들어갔는지확인 : Print(a)

셀 앞부분을 클릭하고 파란색되면 (a : 위에 셀생김 / b : 아래에 ") / select ⊕ X = delete.

문자입력 : ● ● ~~[illegible]~~

• love = 2 b = love Print(b) = 2

• a=1
b=1
c=3
c → run → 3 마지막하나는 Print해줌.
"
a=1 ; b=1 ; c=3 ;

• 한 변수에 여러개 넣는법 → 대괄호 ex) a = [1, 2, 3, 5]
(list) 괄호 없어도 ok (소괄호) tuple

• type(변수) → 어떤 type 인지 ex) int (숫자), list, float (소수점이있는 숫자.), str (문자)

• a = [1, 'love', [1, 'bye']]

• a = { 'a' : 'apple', 'b' : 'banna' } 표제어 : 설명.
↳ type : dictionary)

A=1, b=1 c=a+b
C=2

A= [1,2]
B=[3,4]
C=a[0]+b[0]
C
-4(1+3)

C=a[1]+b[1]
C
-6

A=1.2 a=int(a) print(a) -1
A= '123' print(a[1]) -1. Q. ???
A= 123 print(a[1]) - error ← 이런 index는, 문자나 list에만 可

Index: 내부적인부분

Dict에있는 정보를 가져올때는 pair중의 앞부분을 index의 수단으로 쓴다. 그러면 뒷부분이 나옴.

Type(a[0])

S='abcdef'
N=[100,200,300]

Print s[0] 제일 첫 번째 꺼 .....
        s[-1] 제일 마지막 꺼 .....
S[1:3] = 첫번째에서 세번째의 직전까지 / b,c
1: = 첫번째에서 끝까지
:3 = 첫번째에서 세번째 의 직전까지

Len[ ] = 길이

S.upper( ) - 대문자 됨

R index ??

Strip: 쓸데없는거 없애주는
Split: ' ' -스페이스를 사용해서 나눠라

a= '123' print(a[1]) → 2 (그냥 문자에서도, 두번째꺼)
(two가 아니라 이런 문자처럼.

(   )로 묶으면 tuple, [   ] 로 묶으면 list.

dic → {"a" : "apple" }

print (a ["a"])

S= "abcdef"
0 1 2 3 4 5

1 에서 끝까지 / 0에서 3직전까지

print(s [1:3] , s[1:] , s[:3] , s[:])

bc bcdef abc abcdef.

S.upper()
result = s.find(   ) : 띄어쓰기포함. 몇번째서 시작? ※ 0부터시작
result = s.rindex(?)
print (result)

s= s.strip( ) ← print(s)

띄개로 보기.
tokens = s. split(' ') ← print (tokens)
s = ' '.join (tokens)

s = s.replace ('this', 'that')

앞에 [illegible] 맞게로 보기.

```
a= [1,2,3,4]
for i in a:           # for in : 라는 문법은 in 뒤에 있는 것을 i에 할당하는 것
      print(i)        # for과 if 다음에는 indent를 하는 것이 매우 중요하다.
1234

a=[1,2,3,4]
for i in range(4):   #range는 index를 만들어주는 것 range(4): 0부터 4개 (0,1,2,3)
      print(a[i])        #range(len(a)) 라고 일반적으로 함.
                         # len(a): 그냥 몇갠지
1234
a=['red','green','blue','purple']
for s in a:
      print(s)
red green blue purple

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

a=['red','green','blue','purple']
for i,s in enumerate(a) :                    #enumerate: 숫자를 부여
      print(I,s)
0 red 1 green 2 blue 3 purple

a=['red','green','blue','purple']
b=[0.2, 0.3, 0.1, 0.4]
for i, s in enumerate(a) : #앞에있는게 번호, 뒤에있는게 element. enumerate 없으면 변수 2개 못씀
      print("{}:{}%". format(s, b[i]*100))
 red: 20.0% green:30.0% blue:10.0% purple:40.0%

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=0
if a==0:
    print("yay!")
yay!

if a!=0:
    print("yay!")
else:
    print("no")
no

#시험문제
for i in range(1,3):
    for j in range(3,5):
        print(i*j)
3468

for i in range(1,3):
    print(i)
    for j in range(3,5):
        print(i*j)
1 3 4 2 6 8
순서: 1: 1을프린트, 1을 3,4와 곱함

for i in range(1,3):
    for j in range(3,5):
        if j>=4:
            print(i*j)
4 8
for i in range(1,3):
    if i >= 3:
        for j in range(3,5):
            print(i*j)
아무것도안나옴
```

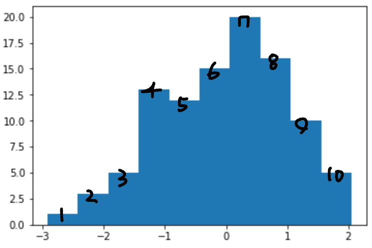
Numpy

```
import numpy as np
import matplotlib.pyplot as plt

np.empty([2,3], dtype='int')  → 2행 3열의 행렬 random
np.zeros([2,3])               → 0으로 구성된 행렬
np.arange(0,10,2, dtype='float64') → 0~10전까지. 2씩증가      array([0., 2., 4., 6., 8.])
np.linspace(0,10,6, dtype=float)   → 0~10까지를 6개로 나눔.   array([ 0.,  2.,  4.,  6.,  8., 10.])

X = np.array([[1,2,3],[4,5,6]]) → 행렬에 들어가는 것들을 직접 쓰는 방법
X.astype(np.float64) → 행렬에 들어가는 것들의 type 바꿈
np.zeros_like(X) → 행렬에 들어가는 것들을 0으로 바꿈

data = np.random.normal(0,1, 100) → 정규분포
print(data)
plt.hist(data, bins=10) → histogram 생성, 바구니가 10개
plt.show()
```



```
X = np.ones([2, 3, 4])
Y = X.reshape(-1, 3, 2)
np.allclose(X.reshape(-1, 3, 2), Y)

                  0~10전까지 숫자들로 구성
a = np.random.randint(0, 10, [2, 3])
b = np.random.random([2, 3])
np.savez("test", a, b)

del a, b
%who  # Print all interactive variables

npzfiles = np.load("test.npz") ) 파일로 저장        a        b
                                                    "        "
npzfiles.files                                   ['arr_0', 'arr_1']

npzfiles['arr_0']

np.savetxt("regression_saved.csv", data, delimiter=",")
!ls -al regression_saved.csv

arr = np.random.random([5,2,3])

print(type(arr))            <class 'numpy.ndarray'>
print(len(arr))             5
print(arr.shape)            (5, 2, 3)
print(arr.ndim) dimension   3
print(arr.size)             30   5X2X3
print(arr.dtype)            float64

a = np.arange(1, 5)         1.2.3.4
b = np.arange(9, 5, -1)     9.8.7.6
print(a - b)                [-8 -6 -4 -2]
print(a * b)                [ 9 16 21 24]

a = np.arange(1, 10).reshape(3,3)     ) 일렬이었던걸 3X3행렬化
b = np.arange(9, 0, -1).reshape(3,3)

   두개다 같음.              세로로 더함                          가로로 더함.
a.sum(), np.sum(a)   a.sum(axis=0), np.sum(a, axis=0)   a.sum(axis=1), np.sum(a, axis=1)
```

In [118]:

입력을 degree로 받지 않고 radiant로 받는다.

```
from matplotlib import pyplot as plt= import matlotlib.pyplot
```

Phasor : sin, cos 의 한 주기

In [160]:

```
# parameter setting
amp = 1           # range [0.0, 1.0]
sr = 10000         # sampling rate, Hz =해상도
dur = 0.5          # in seconds
freq = 100.0        # sine frequency, Hz =1 초에 몇번 왔다갔다
```

In [161]:

```
# generate time 시간의 개념이 있어야 실질적 소리를 만들 수 있음

t = np.arange(1, sr * dur+1)/sr #이런거보고 tic 의개수구하기
```

In [162]:

t=0.0001,0.0002,....0.5000 (duration이 0.5니까 이만큼 하는 것, sr10000이라서 이렇게나눔)

t=np.arange(1,sr+1) 하면 만개가 생기는데 그러면 1초. 그러나 dur이 0.5

t=np.arange(1, sr*dur+1)

이거를 다시 sr로 나누어야 하나의 단위가됨

t=np.arange(1, sr*dur+1)/sr

e-04=1/10000 e-01=1/10

Theta: x축, (0~2파이) s: 결과값 = 총 7개

Theta=np.arrange(0, 2*np.pi, 0.1)/더 빽빽하게 만드려면 여기서 세번째꺼를 더 작게

```
# generate phase
theta = t * 2*np.pi * freq
```

time의 벡터 사이즈= 세타의 벡터 사이즈 (t에다가 뭔가를 곱하기만 한 게 세타니까)

t가 1이고 1에 2파이를 곱하면 그냥 한바퀴도는 것. 거기에다가 몇 바퀴 도는지 정하는 게 freq

```
# generate signal by cosine-phasor
s = np.sin(theta)
```

In [164]:

```
fig = plt.figure()
```

`ax = fig.add_subplot(111)#` Subplot: 221: 2행2열짜리, 1:첫번째 꺼

```
ax.plot(t[0:1000], s[0:1000], '.') # t,s를 1000개만 플라팅:점들의개수총 1000개/[    ]안의 숫자들 달라지면 안됨/ 걍 t,s라고만하면 태극문양안만들어짐
                                          #.대신 -하거나 아무것도없으면 선으로연결
```

Subplot: 221: 2행2열짜리, 1:첫번째 꺼

```
ax.set_xlabel('time (s)') label은 그냥 이름 정하는 것
```

```
ax.set_ylabel('real')
<IPython.core.display.Javascript object>
```

Equidistance= x축에서는 하지만, y축에서는 하지않음.

linear하면(선처럼 생기면, 1차함수) x,y축에서 둘다 equidistance.

```
Text(0, 0.5, 'real')
# generate signal by complex-phasor
c = np.exp(theta*1j)
```

exp가 오일러함수 그위에 `theta*1j`가 들어감. J는 i임

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot(t[0:1000], c.real[0:1000], c.imag[0:1000], '.')
```

```
# a+bi 에서 real: a, imag: b
```

#projection

time은 항상 볼 수 밖에 없음

real만 보려면: 위에서 = cos함수와 유사

imaginary = sin

```
Text(0.5, 0, 'imag')
ipd.Audio(s, rate=sr)
ipd.Audio(, rate=sr : 소리만드는거)1
```

sampling rate: 100Hz 1 초에 100 개의 숫자 표현
1hz 를 표현하는 것 =가능 : 1 번왔다갔다
2hz   = 가능 : 2 번 왔다갔다
10000hz   = 1 초에 만번 왔다갔다. 주어진 숫자는 100 개 - 불가능
Sampling rate 이 충분하게 있어야 표현 가능

처음에는 그냥 모든 주파수대가 똑같음
근데 실제 source 는 gradually decrease 함-그렇게만들어줌
그다음에 formant 를 만들기 위해서 어느 주파수에서 산맥이 만들어지는지 하나하나 표시.

Generate pulse train

```
# generate samples, note conversion to float32 array
F0 = 100; Fend = int(sr/2) #nyquist ; s = np.zeros(len(t))#제일처음의 s, time vector 의 개수만큼 만들면 된다.;
for freq in range(F0, Fend+1, F0):f0 부터 끝까지, f0 만큼 올라감. 여기서는 5000 까지. 50 번 돈다.
    theta = t * 2*np.pi * freq
    tmp = amp * np.sin(theta)
    s = s + tmp 더하고 나면 그 결과가 다시 s 가 됨. 제일처음의 s 가무엇인지 정의 해야 함.
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(t[0:1000], s[0:1000]);
ax.set_xlabel('time (msec)')
ipd.Audio(s, rate=sr) #여러소리를 합친 소리가 난다.
```

#def, function의 이름. 입력할 것들/return: 출력. 그냥 함수를 정의 해 준 것이다.

```
b = np.array([sum(a)])
    return a, b
```

RG: frequency

BWG: 산맥을 만들 때 이 산맥이 얼마나 뚱뚱(width 큼)/홀쭉(width 작음) 한 지

0의 주파수에다가 100정도의 뚱뚱한 산맥을 만들어라

```
#입술을 만들어주는 것.
s = lfilter(np.array([1, -1]), np.array([1]), s)
ipd.Audio(s, rate=sr)
```