ENGL238 English Phonetics

영어영문학과 2018130887 정세은

1)English Consonants, Vowels

-영어의 알파벳은 총 26개로, 자음 21개과 모음 21개로 이루어져 있다. 그 중, 모음은 monophthongs와 diphthongs로 나누어진다.

-모든 소리는 발음할 때 목이 떨리는지 안 떨리는지에 따라 voiced와 voiceless로 나누어진다. 모든 모음은 voiced이다.

-j(y)는 자음이다. year은 자음으로 시작하는 단어, ear은 모음으로 시작하는 단어이다. 헷갈리지 말 것!

2)Phonetics

-Phonology : 인지적, 추상적, 머릿속에서 일어나는 것 vs. Phonetics : 물리적, 실제로 소리 나는 것

-Articulatory phonetics (from mouth) : How to produce speech

-Acoustic phonetics (through air) : How to transmit speech

-Auditory phonetics (to ear) : How to hear speech

3)Articulation

-Vocal track(upper) : lip, teeth, alveolar ridge, hard palate, soft palate(velum), uvula

-Vocal track(lower) : lip, tongue(tip, frond, back, blade, center, root)

-5 speech organs : oro-nasal process(velum), articulatory process(lips, toungue tip, toungue body), phonation process(larynx)

4)Phonation process

-Larynx = voicebox

-voiced : can feel vibration (v,z,l,m,a,i..) -voiceless : can't feel vibration (f,s,k,p,h..)

5)Oro-nasal process in velum

-velum is lowered → nasal sound, when breathing -velum is raised → oral sound

6)Articulatory process: in lips, toungue tip, toungue body

7)Control of constrictios(articulators)

: Each constrictor(lips, tounge tip, tounge body) needs to be more specific in geometry

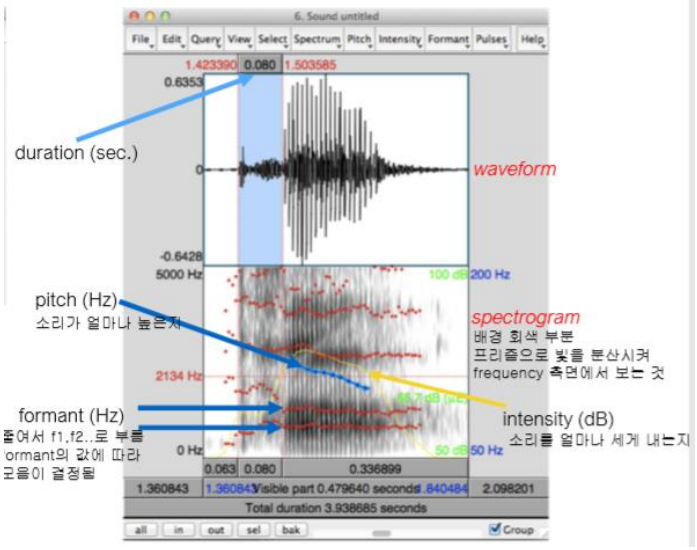
-Constriction location(CL) : lips, tounge body, tip

-Constiction degree(CD) : stops, fricatives, approximants, vowels

8)Phonemes : Individual sounds that form words, a combination of speech organ's actions

<Praat>

1)Acoustics in Praat



-duration : 소리가 지속되는 기간 (초)

-pitch : 소리의 높낮이

(male : 65-200Hz, female : 145-275Hz)

-formant(Hz) : formant의 값에 따라 모음이 결정됨

-intensity(dB) : 소리의 세기

-spectrogram : 배경 회색 부분. 프리즘으로 빛을 분산시켜 frequency 측면에서 보는 것

2)Vowel acoustics

-Measure pitch using Praat : praat 상의 큰 파도 = larynx가 닫혔다가 열리는 떨림. 큰 파도~큰 파도까지 몇 초 걸리는지 계산하고, 1/해당값 하면 Hz를 알 수 있음.

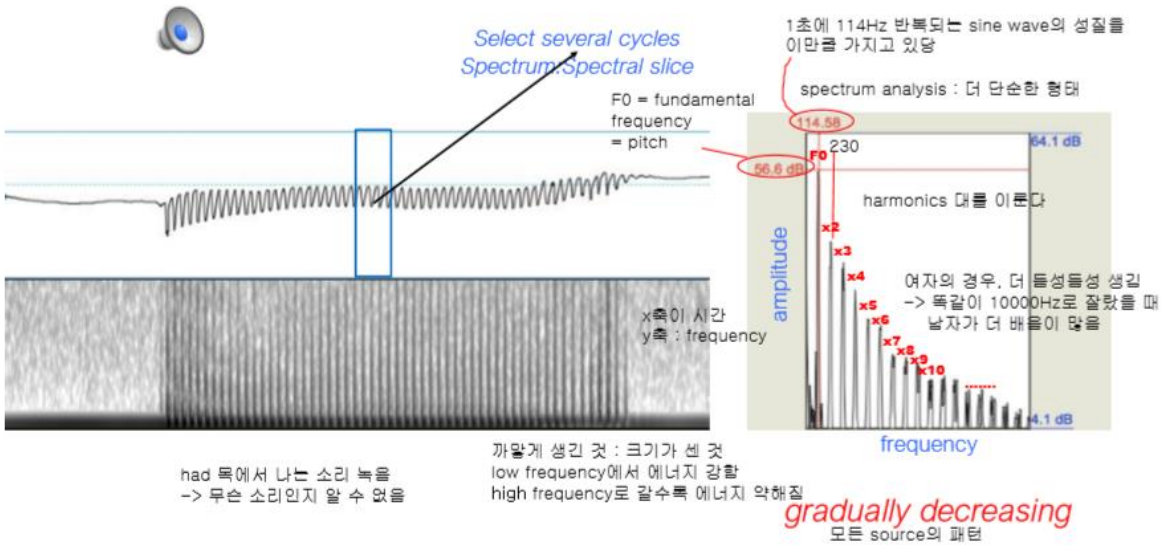
3)Human voice source

-larynx에서 나는 소리. complex tone(pure tone의 총합)임.

-ElectroGlottoGraph(EGG)로 측정됨.

-hamonics로 이루어져 있음.

-가장 낮은 pure tone을 Fundamental frequency(F0)이라고 부름. = 1초당 성대의 개폐가 반복되는 정도



4)Complex tone in spectrum

-모든 signal(sound 포함)은 여러 개의 서로 다른 sine wave의 결합으로 표현됨.

-sine wave: 리드미컬하게 반복되는 기본적인 형태. x축은 시간, y축은 value

-sine wave들을 합하여 새로운 wave를 만들 수 있음. 반복되는 형태는 첫번째 wave(가장 느린 wave)와 같음.

-sine wave를 spectrum 형태로 만들 수 있음.

5)Filter

-vocal tract에 의해 filtered된 소리

-peaks/mountain : vocal tract가 좋아하는 소리 = formants

-valleys : vocal tract가 싫어하는 소리

6)Synthesizing Source

-서로 다른 frequencies와 amplitude로 여러 개의 pure tone을 만든 다음 stereo로 결합하고, 그 다음 mono로 변환시킴.

7)Vowel space

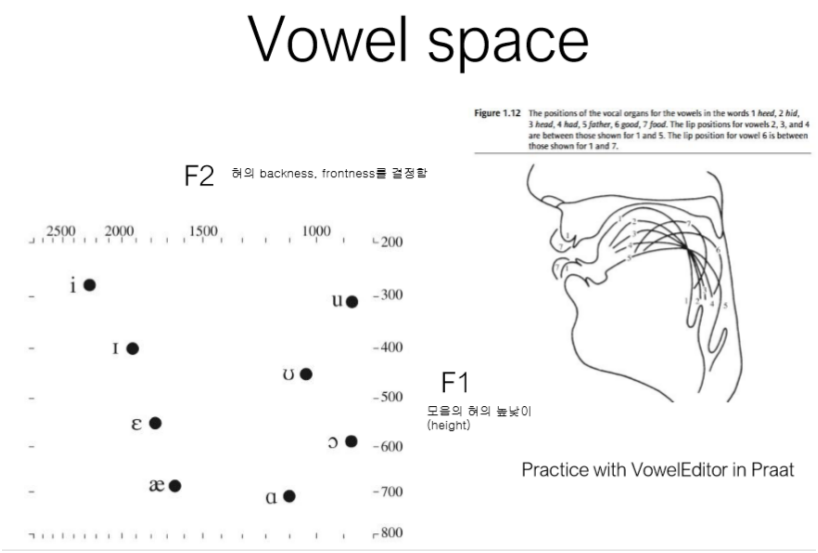


Figure 1.12 The positions of the vocal organs for the vowels in the words 1 *heed*, 2 *hid*, 3 *head*, 4 *had*, 5 *father*, 6 *good*, 7 *food*. The lip positions for vowels 2, 3, and 4 are between those shown for 1 and 5. The lip position for vowel 6 is between those shown for 1 and 7.

언어 = 단어 + 문법

단어: 정보를 담고 있는 그릇

정보를 담는 그릇(단어)이 변수(variable)로서 필요함

프로그래밍의 공통적인 문법

1)변수에 정보를 넣는 것(assign하는 것) variable assignment

2)컨디셔닝에 대한 문법이 필요함. (if문법) if conditioning

3)여러 번 반복 (for문법)

4)함수. 입력을 넣으면 출력이 나오게끔 packaging하는 것. 재사용 가능. 반복적으로 사용

두 개를 숫자를 주면, 처음부터 끝까지 자연수의 합을 구해라.

ex) 7, 10 입력 → 7+8+9+10=34

variable의 종류 : 숫자 or 글자

컴퓨터 프로그래밍에서 =는 equal의 의미가 아님. 오른쪽의 정보를 왼쪽의 variable로 assign한다고 생각하면 됨.

variable은 unique함. 처음에 1을 입력했다가, 다음에 2를 입력하면, 처음 값인 1은 사라짐.

문자를 입력할 때는 반드시 ' ' 안에!

shift + enter : 실행

b : below에 새로운 cell, a : above에 새로운 cell, x : 셀 삭제

```
In [25]: a = 1
         b = 2
         b
         c = 3
         c
Out[25]: 3
```

마지막에 친 variable의 값을 보여줌

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

list로 여러 개 넣을 수 있음.

```
In [29]: type(a)
Out[29]: list
```

무슨 타입인지 알 수 있음

int(1) / list ([1, 2, 3]) / float (1.2) / str ('love')

```
In [41]: a = [1,'love',[1,'bye']]
```

Q. type은? list

속해있는 아이템은? int, str, list

```
In [44]: a = (1,'love',[1,'bye'])
In [45]: type(a)
Out[45]: tuple
```

list = tuple은 같은 이름.

list [ ] 사용, tuple은 ( ) 사용. tuple이 보안에 더 강함.

```
In [46]: a = {'a':'apple', 'b':'banana'}
In [47]: type(a)
Out[47]: dict
```

dictionary. { } 사용. **표제어:설명어** 처럼 쌍으로 항상 이루어져야 함.

```
In [6]: a=[1,2]
        b=[3,4]
        c=a[0]+b[0]
        c
Out[6]: 4
```

a와 b 각각의 list를 만들고, list에서 0번째 값을 가져와서 더할 수 있음.

```
In [10]: a=1; a=float(a); print(type(a))
<class 'float'>
```

a=1인 경우 원래 type은 int이지만, a=float(a)로 입력하여 type을 바꿀 수 있음.

```
In [13]: a=1.2; a=int(a); type(a)
Out[13]: int
```

어떤 variable의 내부 정보로 들어갈 때는 대괄호[ ]를 사용함. 대괄호 안에 들어가는 것은 index를 쓴다.

```
In [14]: a='123'; print(type(a)); print(a[1])
<class 'str'>
2
```

```
In [22]: a={"a":"apple", "b":"orange", "c":2014}
print(type(a))
print(a["a"])
<class 'dict'>
apple
```

dict에서는 pair 중 앞부분을 access의 index로 씀.

<-> 보통은 0번째, 1번째 .. 이런식으로 사용

```
In [23]: a={1:"apple", "b":"orange", "c":2014}
print(type(a))
print(a[1])
<class 'dict'>
apple
```

dict에서 표제어를 str이 아니라 int로 해도 가능.

```
In [30]: s = 'abcdef'
n = [100, 200, 300]
print(s[0], s[5], s[-1], s[-6])
a f f a
```

맨 앞부터 0번째, 1번째 ...

반대로 맨 앞에서부터 반대 방향으로 -1번째, -2번째, ...

제일 첫번째 것과 제일 마지막 거는 일일이 세지 않아도 0번째, -1번째로 찾으면 됨

```
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[:])
a f f a
bc bcdef abc abcdef
```

print(s[1:3]) 은 첫번째에서 3번째의 직전 것(=두번째꺼)까지

print(s[1:] 은 첫번째부터 끝까지

print(s[:3]) 은 0번째부터 3번째의 직전 것(=두번째꺼)까지

```
In [32]: 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]
```

list도 위의 string과 똑 같은 접근방식 사용함.

```
In [33]: len(s)
Out[33]: 6
```

variable의 정보의 길이(length)를 알 수 있는 함수

```
In [36]: s[1]+s[3]+s[4:]*10
Out[36]: 'bdefefefefefefefefefef'
```

값들을 계산하듯이 할 수 있음

```
In [37]: s.upper()
Out[37]: 'ABCDEF'
```

대문자로 바꾸는 함수

variable을 만들고 그 옆에 .을 쓰면 함수가 실행됨

```
In [42]: s=' this is a house built this year.₩n'
         s
Out[42]: ' this is a house built this year.₩n'
```

```
In [43]: result=s.find('house')
         result
Out[43]: 11
```

11번째에서 'house'가 시작됨

```
In [49]: result=s.rindex('this')
         result
Out[49]: 28
```

rindex는 마지막 단어의 위치를 찾아줌

```
In [51]: s = s.strip()
         s
Out[51]: 'this is a house built this year.'
```

불필요한 것들을 제거하고 순수한 텍스트만 남겨주는 함수

```
In [52]: tokens = s.split(' ')
         tokens
Out[52]: ['this', 'is', 'a', 'house', 'built', 'this', 'year.']
```

긴 string을 스페이스를 기준으로 잘라서 단어들로 모아서 list 만들 수 있는 함수

```
In [54]: tokens = s.split(',')
         tokens
Out[54]: [' this is a house', ' built this year.₩n']
```

,을 기준으로 자를 수도 있음.

```
In [60]: s = ' '.join(tokens)
         s
Out[60]: ' this is a house  built this year.₩n'
```

token에 있는 잘라진 단어들을 붙일 수 있음

```
In [65]: s=','.join(tokens)
         s
Out[65]: 'this,is,a,house,,built,this,year.'
```

```
In [64]: s=s.replace('this', 'that')
         s
Out[64]: 'that is a house  built that year.'
```

모든 this를 that으로 바꿔라.

#을 앞에 붙이면 실행이 되지 않고 note로 남게 됨.

또는 Markdown ▼

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

for i in a

print(i)

in 뒤에 있는 것(a)을 하나씩 불러서 i가 그것을 행하고 무언가(print)를 해라.

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

range 뒤에 숫자가 나오면 list를 만들어 줌.

ex) range(4)는 4개의 index를 만들어 줌. (0부터 3까지)

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%
```

enumerate(a)는 a의 variable에 번호를 매김.

(i,s)=(0,red) (1,green) …

print("{}:{}%".format(s,b[i]*100)) 은

{}:{}%를 출력하는데, (s,b[i]*100)의 형태로 출력함

In [17]:
```
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%
```

len(a)와 len(b)는 같아야 함.

zip→a와 b가 각각 pair로 연결됨

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

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

a=0이라면 "yay!"를 출력하라.

a=0이라면 "let's go"를 출력하라.

In [35]:
```
a = 0
if a < 0:
    print("yay!")
    print("let's go")
else:
    print("no")
```

```
no
```

<= 또는 >=처럼 부등호가 먼저 나와야 함.

In [36]:
```
for i in range(1,3):
    for j in range(3,5):
        print(i*j)
```

```
3
4
6
8
```

range(1,3)은 1,2,3이 아니라 1,2

첫번째 루프는 1,2이니까 두 번 돌고
두번째 루프는 3,4이니까 두 번 돎.

→ 총 4번 돌게 됨.