음성학 summary 과제

음성학(phonetics) : 사람의 **말에 관한 모든 연구**

cf) 음운론(phonology) : 음성(sound system)에 관한 연구 / 음성이 우리의 머릿속에 들어가는 **인지적** 절차에 관한 연구

- 음성학은 음운론보다 더 깊게 들어가서 사람이 내뱉는 음성의 **물리적 차이를** 연구

음성학은 3가지로 나눠진다.

1. Articulatory phonetics
   - 사람의 원리(입 모양 / 턱의 변화 / 혀의 변화 / 성대 운동)를 통한 조음 현상에 관한 음성학적 연구
   - 가장 원리적인 연구
2. Acoustic phonetics
   - 공기와 소리가 어떻게 물리적으로 음성이 되는가에 관한 음성학적 연구
3. Auditory phonetics
   - 소리가 귀로 들어와서 어떻게 우리가 음성을 듣는가에 관한 음성학적 연구

이번 수업시간에서 주로 다룰 음성학 분야는 Articulatory phonetics

Articulatory phonetics는 음소의 조음 과정을 연구하는 것이 주목적

여기서 **Phoneme(음소)이**란

- 어떤 언어에서 의미 구별 기능을 갖는 음성 상의 최소 단위 혹은 개별적인 소리
- 음소는 **자음(consonant)**과 **모음(vowel)**으로 구성
- 예를 들어 'psycho'라는 단어는 /s aɪ k oʊ/라는 음소로 구성되어 있으며 여기서 중요한 점은 철자≠ 발음이라는 것이다.

Articulatory phonetics는 5가지 요소로 **음소**를 판단한다.

1) Constrictor=articulators=5 speech organs
2) Velum(연구개) -> up or down
3) Larynx(후두/성대) -> open or closed
4) Constrictor location(CL)
5) Constrictor degree(CD)

1) Constrictor = articulators = 5 speech organs
   a. lips
   b. tounge tip
   c. tounge body

   → articulatory process : 입 모양, 혀의 변화 등으로 음소를 나누는 가장 기본적인 process

   d. velum(=soft palate) --> oro-nasal process : 비음과 비음이 아닌 음소를 나누는 process
   e. larynx(=voicebox) --> phonation process : 무성음과 유성음을 나누는 process
2) Velum(연구개) -> up or down
   - Velum이 up(off) -> 비음이 아닌 것 ex) p,b,t,d,.....
   - Velum이 down(on) -> 비음(nasal) ex) m,n,ŋ
3) Larynx(후두/성대)
   - Larynx이 opened(on) -> 무성음(voiceless) ex) p,f,θ,t,s,ʃ,k,h
   - Larynx이 closed(off) -> 유성음(voiced) ex) v.z,l,m,a,i,...
4) Constrictor* location(CL)
   * 여기서 constrictor는 lip/ tounge tip / tounge body 세 가지다
   - constrictor의 위치는 어디인가로 판단(앞 or 뒤)
   *** 모음은 constrictor로서 tougue body만을 쓴다.**

   자음만을 분류했을 때 <참고 1>
5) Constrictor* degree(CD)
   * 여기서 constrictor는 lip/ tongue tip / tongue body 세 가지다
   - constrictor가 얼마나 수축하였는지로 판단 (위 or 아래)

   *** 모음은 막힘이 없어서 degree가 낮다 -> 다만 자음 h,ɾ,ɻ은 여기에 속한다.**

자음만을 분류했을 때<참고 2>

Praat : 음성 분석 프로그램

<참조3>

Praat를 통한 음성 분석 요소

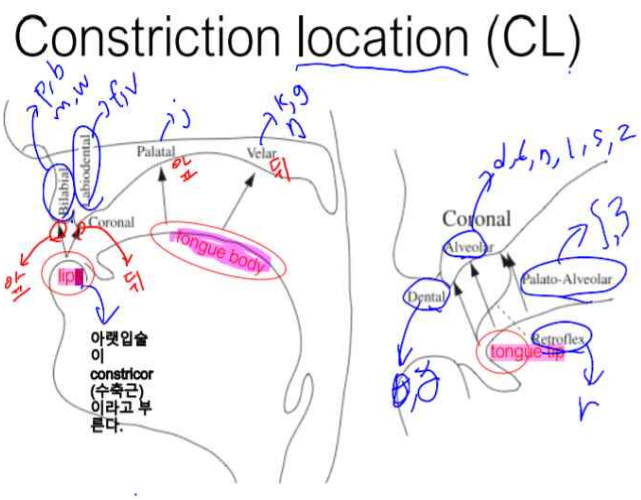
1. duration : 시간을 나타낸다. / 단위는 sec.
2. intensity : 음의 세기를 알려준다 / 단위는 DB(데시벨) / 노란색 선으로 표시
3. pitch : 음의 높낮이를 알려준다 / 단위는 Hz(헤르츠) / 파란색 선으로 표시

   -> 남성과 여성은 일반적으로 pitch range가 다르므로 성에 따라 다르게 pitch range를 설정해야 한다. 65-200 : 남성 / 145-275 : 여성

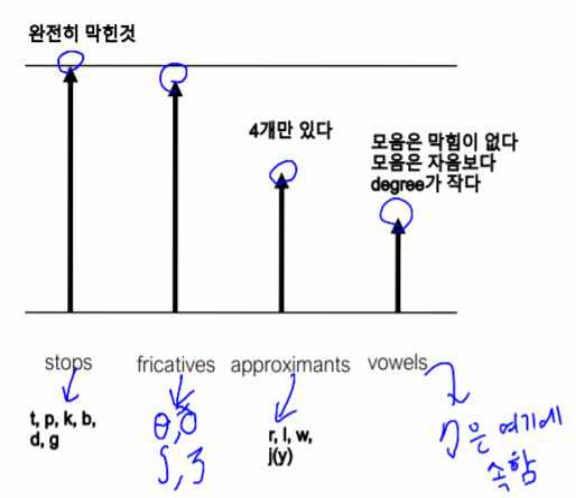
   -> 여기서 중요한 점은 시간은 이론상 무한대까지 쪼갤 수 있으므로 소리의 값이라는 수치도 무한대이다. 따라서, 'Hz'같이 1초 동안 몇 번 진동했는지를 나타내는 수치가 필요하다. 예를 들어 145Hz는 1초 동안 145번 진동한다는 뜻이고 이는 또한 1/145Hz를 해서 0.00689초에 한 번 진동한다는 것을 알 수 있다.

   -> 또한, pitch와 intensity는 독립적이다.
4. formant : 모음이 무엇인지 결정하는 요소 / 단위는 Hz(헤르츠) / 빨간색으로 표시된다.

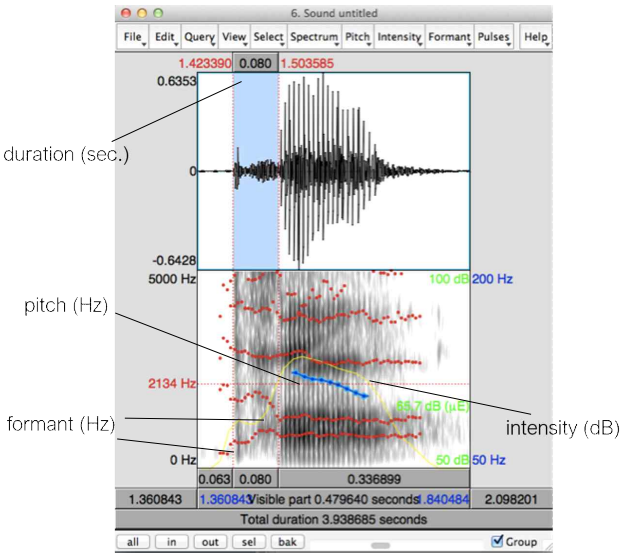
   ex) F1 1000, F2 1500 ->이것이 무슨 소리인지 혹은 모음인지를 알 수 있다.



참조 1



참조 2



참조 3

문제 예를 들어서

/b/라는 발음이면

	Cl	cd
Lips	bl	stop
Tt		
Tb		
Velum raised		
Larynx open		

/b/ -> /d/
Tt aveolar stop

/d/ -> /z/
Stop -> fricative

/n/ fricative -> stop
Velum Lower

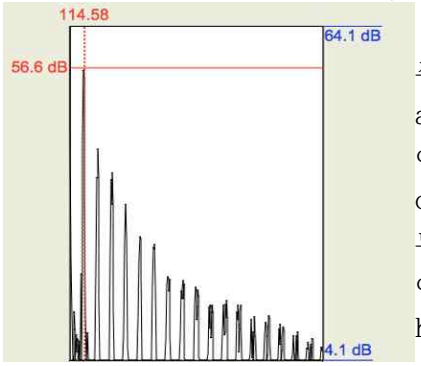
여기서 larynx을 오픈하면(숨쉬는 것과 비슷하다고 생각하면 된다.)
/ŋ/

Hz(주파수) : 1초에 반복된 vocal folds의 진동
일정 주기로 반복되는 함수 = **sine wave (: 0에서 시작하여 일정 주기로 반복하는 함수)**
그러면, 여기서 생각해 볼 수 있는 게 우리의 목소리를 **sine wave로 표현**할 수 있지 않냐는 것
실제로 Pratt를 통해 sine wave를 만들 수 있다. Pratt를 통해 sine wave를 만들려면 pure tone을 만들면 된다. 즉, sine wave = pure tone이라는 뜻
cf) - Sampling Frequency :1초 동안 어느 정도의 정보를 줄 것인가?
- Tone Frequency는 sine wave의 주파수
한 주기를 선택하면 한 주기의 duration = 1초 / 주파수(Hz / tone frequency)가 되는 것을 알 수 있다. 그리고 또한 pitch = tone frequency라는 것도 알 수 있다.
이 pure tone을 spectra slice를 하면 x축을 주파수로 하는 그래프(Sampling Frequency에 의해 결정되는 것 이것의 2분의 1이 주파수의 range)가 나오는데 이는 어떤 주파수 때의 성분(에너지)가 가장 많은지 그 intensity를 보여준다. 그리고 가장 높은 intensity를 가지는 주파수는 amplitude가 가장 크다고도 할 수 있다. 따라서 y축은 amplitude가 된다.
ex) tone frequency를 440 / amplitude를 1로 설정하면(최소 -1 ~ 최대 1)
sine graph가 나오게 되고 한 주기를 선택하면 0.02…. 로 1을 440으로 나눈 값이 되고 또한 이를 spectra slice 하면 440Hz에서 가장 높은 intensity, amplitude를 보여준다.

결론 : 음성학적 sine wave <x축 : 시간 / y축 : voltage>이고 이를 x축을 frequency로 y축을 amplitude로 변환한 것이 spectrum이다. 그리고 이 하나의 sine wave는 하나의 소리를 나타낸다. 이는 곧 모든 신호는 다른 여러 개의 sine wave의 결합(=complex tone)으로 설명할 수 있고 이는 또 모든 신호를 단순한 sine wave로 쪼갤 수 있다는 뜻이 된다.

sine wave의 합 = complex tone의 예시
100Hz 큰 소리 / 200Hz 작은 소리 / 300Hz 중간 소리 세 개의 sine wave를 합하면 complex tone의 그래프가 나오는데 여기서 complex tone의 특징은 제일 낮은 주파수와 반복 주기가 똑같다는 점이다. 또 이를 보면 주파수가 일정 비율로 증가하는데 이를 harmonics라고 하며 가장 낮은 주파수를 fudamental frequency(=f0)라고 한다.
사람이 내는 음성은 이러한 harmonics라는 규칙성을 보이는 데 그중 가장 중요한 것이 human voice source(사람의 larynx에서 나는 소리)이다. (이를 측정할 때 EGG라는 장비를 사용, 양쪽에 전기를 쏴서 성대가 붙으면 전기가 통하는 원리를 이용)

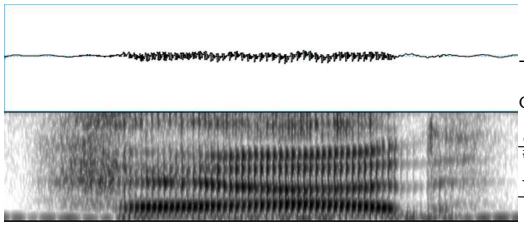
Human Voice Source의 스펙트럼화



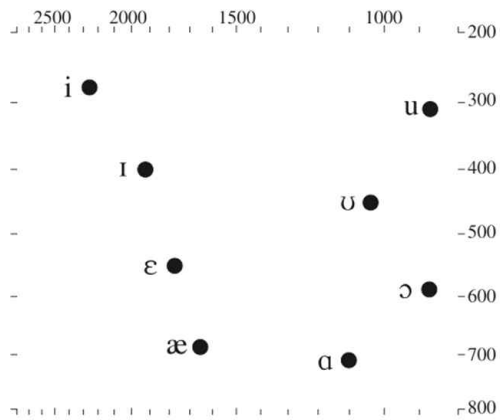
이를 보면 가장 낮은 주파수인 114.58을 f0라고 하며 이는 주파수가 배음하고 있고 또한 배음 관계의 주파수의 amplitude가 gradually decrease 한다는 것을 알 수 있는데 이것이 human voice source의 특징이다.
cf) 남자와 여자의 간격은 누가 더 넓을까? (=harmonics는 누가 더 적을까?)
여자는 amplitude / pitch가 더 높으므로 간격이 더 넓고 harmonics는 더 적다.)

성대에서 나는 human voice source는 filter(tube에서 입모양/혀위치 등의 변화)에 의해 소리가 달라진다. 이러한 filter를 거치면 배음의 구조는 유지되지만, amplitude가 gradually decrease 하던 구조가 깨지고 서로 다른 complex tone이 된다. -> soure는 gradually decrease의 amplitude를 filter는 curved 한 amplitude를 가지고 있다.

source와 filter를 통해 나온 소리를 x축을 시간 / y축을 주파수 / z축을 amplitude로 하여 시각화 한 것이 spectogram이다.



-> 여기서 amplitude가 높을수록 검은색이 더 짙어지는 데 이 짙은 곳을 peak이라고 하며 옅은 곳을 vally라고 하며 이둘이 하나의 산맥을 형성한다고 하여 이를 mountain이라고도 한다. 이러한 peak과 valley를 나눠주는 것이 바로 filter인데 왜냐하면 filter 별로 각각의 주파수를 강하게 하거나 약하게 하는 성질을 가지고 있기 때문이다. 즉, 그 filter에 가장 적합한 소리가 바로 우리가 듣는 소리라는 뜻이다. 또한, f0는 source의 부분이며 mountain과는 관련 없다. 그리고 mountain 중에서 첫 번째 mountain을 F1, 두 번째 mountain F2라고 하며 F1, F2를 합쳐서 Formant이라고 한다. 이 두 산맥만 있으면 우리는 소리를 구별할 수 있으므로 formant를 이용하여 우리 혹은 인공지능도 모음을 구별할 수 있다.



마지막으로 F1과 F2가 위치할 때 이는 우리의 입모양과 비슷한데 여기서 F1은 혀의 높낮이를 F2는 front와 back을 결정하여 하나의 모음 sound를 끌어낸다.

10/02

코딩 = 자동화

왜 자동화를 할까요?

똑같은 일이 반복되는 경우가 많으므로 (예를 들어 자동차 산업) 매번 할 때마다 반복하지 않기 위해 -> 반복되는 일의 자동화

코딩 -> 핸드폰에 있는 모든 것 / 컴퓨터의 모든 것 모두 자동화로 이루어져 있다.

프로그래밍 언어

프로그래밍 언어에 대해 이해하기 전에 언어에 대한 이해가 필요하다

모든 언어의 특징 : 단어와 문법이 있다.

커뮤니케이션 : 단어의 combine으로 정보를 전달

단어는 뭘까? -> 정보를 담고 있는 그릇

문장을 전달할 때 어떤 단어를 선택할 것인가 -> 내가 이야기하고 싶은 완전한 의미를 전달 가능

이것과 똑같이 프로그래밍 언어도 생각할 수 있다

프로그래밍 언어에서 단어 : 정보의 그릇 -> 즉, 변수 (variable)

이 정보를 가지고 기계와 communication을 하려면 문법이 필요

문법의 기본 구조

1. 정보를 변수에 assign한다.
2. conditioning에 대한 문법 -> <if>문법을 사용
3. 여러번 반보 -> <for / loop> 라는 문법을 사용

이 세 가지가 프로그래밍 언어에서 가장 필요한 개별적인 문법

이 개별적인 문법을 익힌 뒤에는 함수를 배워야 소통이 가능하다.

4. 함수 -> 문법에서 가장 중요

함수 : 입력(input / praat에서는 마우스의 클릭) -> 출력(output / praat에서 소리의 출력)

함수 : if / for 등의 문법을 packaging한 것

ex) 두 개의 자연수를 주면 시작부터 끝까지 자연수 만큼의 증가로 총합을 구해라

--> 이를 기계가 이해하게 만드는 언어 -> 바로 프로그래밍 언어

7 ~ 10 -> 34 등의 갑

함수의 특징

1. 재사용 2. 반복가능

Python의 사용

a = 1 -> 오른쪽에 있는 정보를 왼쪽으로 assign한다

print(a)를 쓰면 1이라는 값이 나오는데 여기서 print는 누가 만들어놓은 함수이다

어떤 변수를 그 안에 넣으면 screen out의 역할을 하는 함수다.

그리고 anaconda는 함수들의 집합(패키지)으로 여러 가지 함수를 활용하여 코딩을 할 수 있게 한다.

문자라고 인식하게 하려면 “” 혹은 ‘’을 써야한다 .

같은 변수면 그 변수에 덮어 쓴다
; 은 칸을 나눠 준다(구문이 끝났음을 알려준다)

[] 한꺼번에 담아준다(list) -> list로서 담아준다 이것이 list인지 아닌지 구별하는 방법 -> 이것의 함수가 type (어떤 형태인지 알려주는 함수)

type 함수의 결과값

a = 1 -> int 정수형의 약어

a = 1.2 -> float 실수형의 약어

a = “love”-> string 글자의 Unicode 코드로 이루어진 불변한 순서있는 집합

list안에는 반드시 숫자만 들어갈 필요가 없다 “문자”가 들어갈 수 있다 또 list 속에 list가 들어갈 수 있다 . 모든 list는 []로 쓰지만 ()을 써도 된다 list와 tuple은 완전 똑같다

but, tuple이 조금더 보안에 강하다 -> 바꾸기가 힘들다

, 는 정보를 나눠 준다

dictionary {}을 쓴다 -> dictionary는 immutable한 키(key)와 mutable한 값(value)으로 맵핑되어 있는 순서가 없는 집합

variable 복습

정보에 해당되는 부분이 두 가지 숫자 / 문자

칸이 달라지면 밑에 있는 정보로 overwirte 가 된다.

Variable 에 정보가 들어가 있다고 할 때 어떻게

A = 1; a = float(a); print(type(a))

Float = 정수에서 소수점이 있는 것을 다시 a 에 넣어줌

A= 1.2; a = int(a); print(type(a))

Int = 정수

[x] 여기서 x 는 Index 를 쓴다 -> 이를 통해 내부적인 정보를 얻을 수 있다

X 는 0 부터 시작한다.

a = '123'; print(type(a)); print(a[1])

123 -> 하나로 인식 '123'-> 여러 개로 인식

이를 리스트로 만들어도 여러 개로 인식

Pair 앞부분을 index 로 쓴다

그냥 에서는 0123..을썼다

But, dict 에서는 앞부분의 a,b,c 를 index 로 쓴다

["a" : apple, "b" : "orange"]

표제어로 접근한뒤 내용으로 접근하는 것과 동일한 방식

a = {1 : "apple", "b" : "orange", "c" : 2014}

print(type(a))

print(a[1])

이를 보면 string 이 아닌 숫자로도 가능하다

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

print(a[0])

print(type(a[0]))

type 은 tuple 이고 0 번쨰는 (1,2,3)으로 나온다

string 하고 list 에서 정보를 접근하는 방식이 same

s = 'abcdef'

n = [100, 200, 300]

print(s[0], s[5], s[-1], s[-6])

-1 번째 하고 -6 번째

-1 은 0 번째에서 -1 번째 / 0 에서 -6 번째

즉 0 을 기준으로 왼쪽으로 counting

제일 첫번째는 0 마지막은 -1 로 적으면 된다

➔ 따라서 개수에 대한 counting 이 필요 없어진다.

s = 'abcdef'
n = [100, 200, 300]
print(s[0], s[5], s[-1], s[-6])
print(s[1:3], s[1:], s[:3], s[:])
print(n[1:2], n[1:], n[:2], n[:])
[1:3]이면 첫번째에서 두번째 까지만 가지고 온다
끝은 그 전 까지로 명시한다
(처음은 0 으로 시작해야 한다.)
이를 보면 string 과 list 는 같은 정보 접근방식을 가지고 있다는 점을 알 수 있다 .

Len()-> 정보의 길이를 알려준다

s[1]+s[3**]+**s[4:]*10
'bdefefefefefefefefef'
하면 string 도 문자의 개수로 연산가능하다는 점을 알 수 있다

x. upper()
x 라는 variable 를 대문자로 만들어 준다.

String 이 왜 중요한지?
Text 분석이 중요하기 때문
text 분석의 기본이 string 이다

if string = ' this is a house built this year.₩n'

s.found('단어')
-> 이 단어가 나오는 순번을 알려준다 -> 몇 번째인지 알려준다.
그리고 제일 첫번째 나오는 index 만 찾아준다
Ex) this 가 두 개여도 첫번째 this 의 순번만 알려준다
Result = s.found('this')
1

s.rindex('단어')
마지막에 나오는 index 즉, 왼쪽방향으로 count 해서 처음 나오는 단어의 오른쪽 방향의 순서를 알려준다.
R = right
Ex) result = s.rindex('this'); result
23

s.strip 은 순수한 텍스트만 남겨주는 함수
s = s.strip(); s

'this is a house built this year.'
앞의 공백과 ₩n 을 없애준다

tokens = s.split(" "); tokens
문장을 단어의 리스트로 만들어 주는 함수
s 라는 string 을 split 함수의 입력을 이용해서 잘라라.
' '(space) -> 을 이용해서 잘라라.
's' -> s 를 기준으로 잘라라.

Ex) tokens = s.split(" "); tokens
['this', 'is', 'a', 'house', 'built', 'this', 'year.']

join : 문자열을 합치는데 사용합니다. 구분자가 앞에서 사용되어집니다.
tokens = " ".join(s)
't h i s i s a h o u s e b u i l t t h i s y e a r .' 앞에서의 split 된 string 을 붙여준다

s = s.replace('this', 'that') # replace instances of t with u inside s

s

replace 해준다

과제를 .ipynb 형식으로 직접노트

#을 통해 명령어를 실행하지 않을 수 있고 또한 형식을 code 에서 markdown 으로 바꿔서 실행이 안되게 할 수 있다. 이 상태에서 #을 붙이면 강조된다.

10 / 10일 배운 거

in 뒤에 있는 것을 하나씩 돌려서 i가 그 하나 씩 받아서 < > 실행하라 a에 있는 것을 하나씩 불러서 i에 할당하고 이를 print 해라

In [5]:

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

```
1
2
3
4
```

In [17]:

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

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

range뒤에 함수가 오면 어떤 list를 만들어 준다 range(x)면 x개의 index를 만들어줘라 라는 뜻 len (x) range 함수 첫번째는 0, 1, 2, 3, ...

In [19]:

```
a = ['red', 'green', 'blue', 'purple']
print(a[0]); print(a[1]); print(a[2]); print(a[3]);
# 이것이 각각 print해주는 자동화가 없는 방법
for i in a:
    print(i)
# 위에 거와 똑같은 결과값을 보여주지만 훨씬 간단하다.
```

```
red
green
blue
purple
red
green
blue
purple
```

In [25]:

```
a = ['red', 'green', 'blue', 'purple']
for i in range(4):
    print(a[i])
for i in range(len(a)):
    print(a[i])
for i in range(len(a)):
    print(i)
```

```
red
green
blue
purple
red
```

```
green
blue
purple
0
1
2
3
```

In [31]:

```
a = ['red', 'green', 'blue', 'purple']
b = [0.2, 0.3, 0.1, 0.4]
for i, s in enumerate(a):
    print(i); print(s); print(a[i])
# enumerate 함수 :리스트에 번호를 매긴다 / 원래 순서가 있는데 여기에 번호를 추가로 매겨준다. / 번호가 있기때문에 변수를 두개 받아줘야한다.
                    # i는 번호 / s는 list의 자기자신
```

```
0
red
red
1
green
green
2
blue
blue
3
purple
purple
```

In [38]:

```
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))
    # ("{} : {}%".format(s, b[i]*100)) s 에서는
    # a에서의 'red', 'green', 'blue', 'purple'을 받아주고 (','로 구별해 준다)
    # b[i] b에서 첫번째, 두번째, 세번째, 네번째를 받아주고 /
    # "{} : {}%".format은 이러한 형식으로 print out하고 앞부분은 {} 앞 , 뒷부분은 {}% 뒤로 받아준다, 중간은 :
```

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

In [44]:

```
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))
    # zip은 두개를 따로 따로 합치는 함수 / 하나로 합쳐져서 a가 s에 / b가 i에 들어간다
```

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

In [51]:

```
a = 123
b = 0
if a == 0:
    print("yay!")
    print("let's go")
# if에서 =은 '=='로 표현된다 -> 여기서는 a-0이면 yay!를 print해라
```

```
if a != 0:
    print("김상준")
# if에서 0이 아니면 표현 -> !=
# 작거나 같으면 >=, 크거나 같으면 <=으로 표현
else:
    print("no")
# else = if not
```

김상준

In [52]:

```
a = 123
if a == 0:
    print("yay!")
else:
    print("no")
    # else = if not
```

no

In [2]:

```
for i in range(1,3):
    for j in range(3,5):
          if j >=4:  # j=3일때 x -> 4번이 아니라 두번만 된다.
            print(i*j)
    # 시험에 나온다 !! 중요 !!
    # range은 마지막의 전까지 / (1,3) -> 1,2
    # for의 뒤부분으로 가야지 print가 실행된다.
    # for -> for -> if 이렇게 가면 뒷부분으로 가지 않으면 error가 난다.
    # 칸 띄우기가 중요하다
```

4
8

In [ ]:

```
a = [1,2,3,4]
```